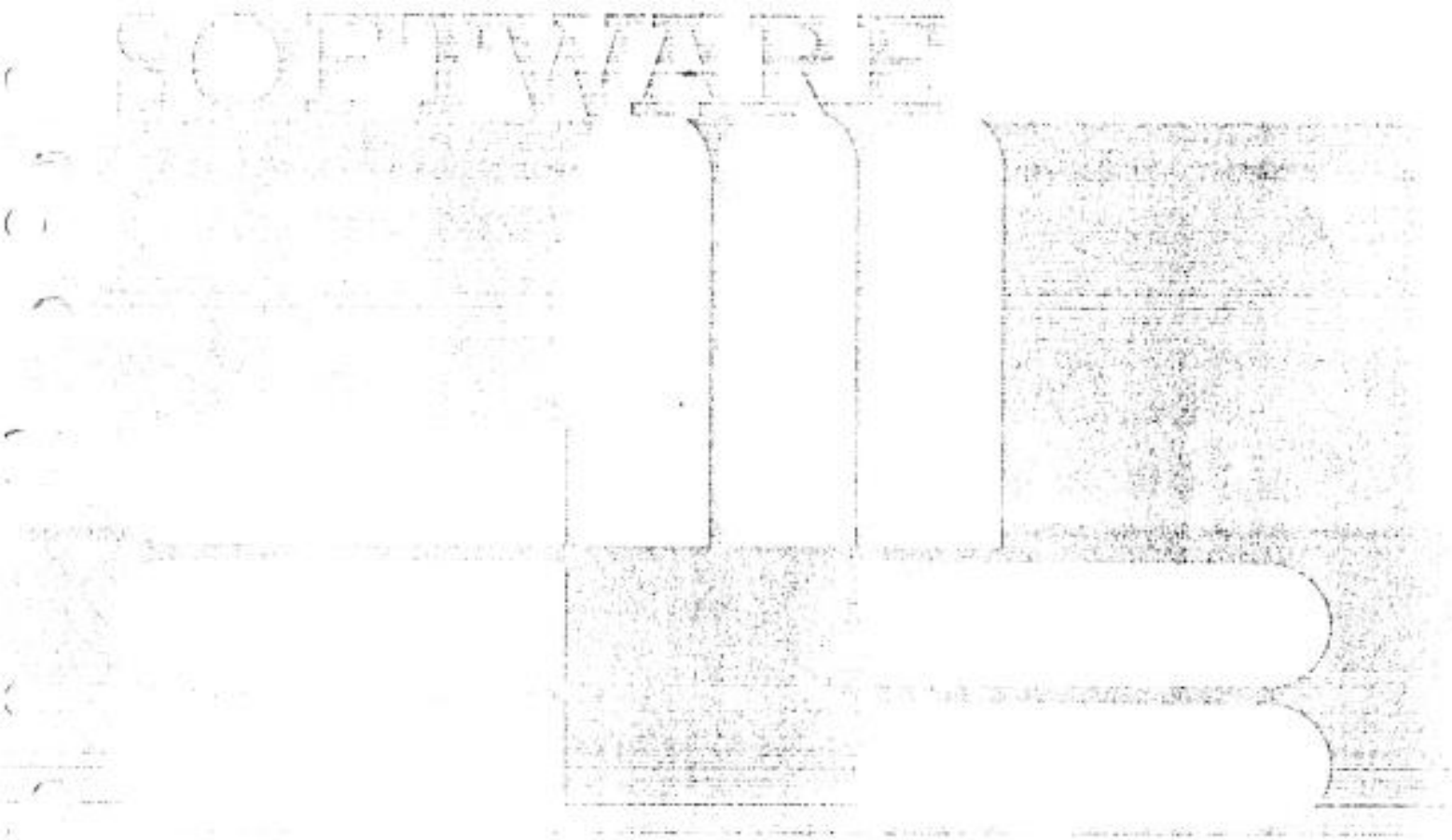
MODCOMP

MAX IV and MAX 32 Programmer's Reference Manual

Task/Overlay Cataloger

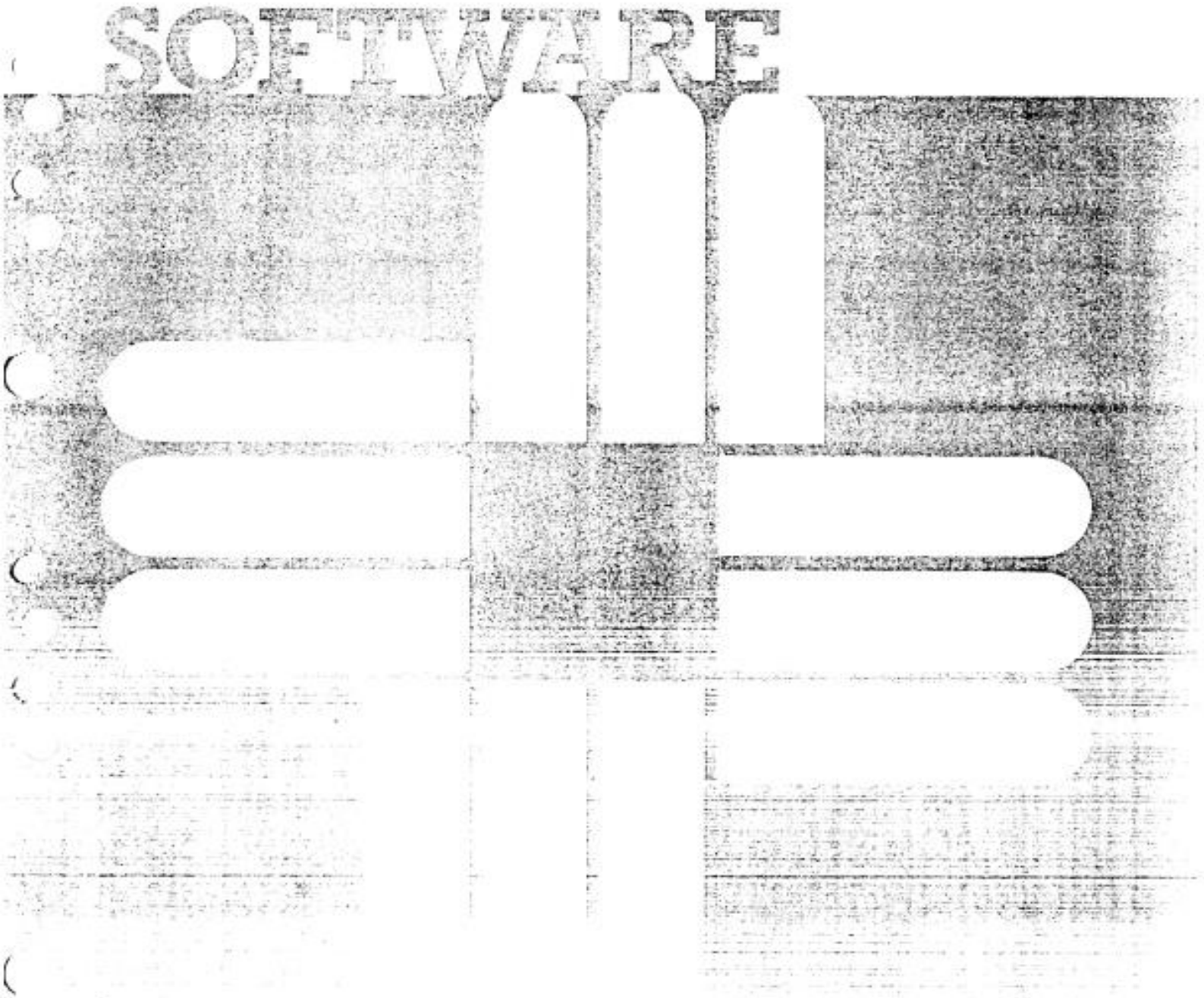


211-804011-L01

MODULAR COMPUTER SYSTEMS, INC

MAX IV and MAX 32 Programmer's Reference Manual

Task/Overlay Cataloger



MANUAL HISTORY

Manual Order Number: 211-804011-L01
Title: MAX IV and MAX 32 Task/Overlay Cataloger, Programmer's Reference Manual
Product Number: 600404-106L.0 and 612404-106A.0

REVISION LEVEL	DATE ISSUED	DESCRIPTION
000	08/76	Initial Issue.
--	05/77	Reissue (A.3).
--	01/78	Reissue (G.0).
--	05/78	Reissue (H.0).
--	02/79	Reissue (I.0).
--	07/80	Reissue (J.0).
--	04/81	Reissue (J.1).
--	08/81	Reissue (J.2).
J01	05/77	Reissue (J.2). Changed the Manual Order Number from 210-610500-021 to 211-804011-J00.
J02	10/82	Revision (J.3).
K00	05/83	Reissue (K.0).
L00	09/84	Reissue (L.0). Supports two new directives: SQU and CLE. Drops support of another: CHA.
L01	05/85	Reissue (L.0 & A.0). Supports MAX IV revision I.0 and MAX 32 revision A.0.

Contents subject to change without notice.

Copyright ©1976, by Modular Computer Systems, Inc.
All Rights Reserved.
Printed in the United States of America.

PROPRIETARY INFORMATION

THIS MANUAL CONTAINS CONFIDENTIAL PROPRIETARY INFORMATION PROTECTED BY SOFTWARE PROPERTY RIGHTS AND IS MADE AVAILABLE UPON THE CONDITION THAT THE SOFTWARE CONTAINED HEREIN WILL BE HELD IN ABSOLUTE CONFIDENCE AND MAY NOT BE DISCLOSED IN WHOLE OR IN PART TO OTHERS WITHOUT THE PRIOR WRITTEN PERMISSION OF MODULAR COMPUTER SYSTEMS, INC.

FOR GOVERNMENT USE THE FOLLOWING SHALL APPLY

RESTRICTED RIGHTS LEGEND

USE, DUPLICATION OR DISCLOSURE BY THE GOVERNMENT IS SUBJECT TO RESTRICTIONS AS SET FORTH IN PARAGRAPH (b) (3) (B) OF THE RIGHTS IN TECHNICAL DATA AND COMPUTER SOFTWARE CLAUSE IN DAR7-104.9(a) (OR SUCH EQUIVALENT FPR, NASPR, AND/OR DOEPR CLAUSE AS MAY BE APPLICABLE).

MODULAR COMPUTER SYSTEMS, INC
1650 WEST McNAB ROAD
FORT LAUDERDALE, FL 33309

PREFACE

Audience
This manual is directed to system operators and system programmers.

Subject
This manual describes the MAX IV and MAX 32 Task/Overlay Catalogers (TOC and TOC32). It provides information about: directory handling, file handling, module handling, and resource defining. Descriptions of the TOC/TOC32 directives and error messages are included.

Product Support
TOC requires the support of the MAX IV (Model 8240) Operating System and TOC32 requires the support of the MAX 32 (Model 8250A) Operating System.

Special Symbols and Notations
A revision bar (|) located in the margin of the page in the text indicates a change to the manual. This change generally represents a technical change to the product due to product revision. A revision bar is also entered in the Table of Contents to flag the general location of changes in the text.

Related Publications
Refer to the manuals listed below for additional information. When ordering manuals, use the manual order number listed. The latest revision (REV) will be shipped.

Manual Order Number	Title
211-804010-REV	MAX IV and MAX 32 LINK EDITOR (M4EDIT and LNK32) Programmer's Reference Manual (PRM)
212-804001-REV	MAX IV DATA STRUCTURES System Design Manual (SDM)
213-838001-REV	MAX 32 DATA STRUCTURES System Design Manual (SDM)
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM System Guide Manual (SGM)
213-838001-REV	MAX 32 GENERAL OPERATING SYSTEM System Guide Manual (SGM)
213-804005-REV	MAX IV BASIC I/O SYSTEM System Guide Manual (SGM)
213-804005-REV	MAX 32 BASIC I/O SYSTEM System Guide Manual (SGM)
213-804006-REV	MAX IV UNIT RECORD DEVICE HANDLERS System Guide Manual (SGM)

213-838006-REV	MAX 32 UNIT RECORD DEVICE HANDLERS System Guide Manual (SGM)
213-804007-REV	MAX IV DATA STORAGE DEVICE HANDLERS System Guide Manual (SGM)
213-838007-REV	MAX 32 DATA STORAGE DEVICE HANDLERS System Guide Manual (SGM)
213-804008-REV	MAX IV COMMUNICATION HANDLERS System Guide Manual (SGM)
213-838008-REV	MAX 32 COMMUNICATION HANDLERS System Guide Manual (SGM)

TOC REVISION L.0 and A.0 SUMMARY

NOTICE

One important reminder that the reader should be aware of in this release of the TOC manual is the fact that this is a common manual for both MAX IV and MAX 32 users. References to "TOC" also refer to "TOC32" unless specific differences are called out under the individual commands. TOC32 is used for the preparation of 32-bit tasks or overlays for the MAX32 Operating System.

TOC REVISION L.0 SUMMARY

Revision L.0 of the MAX IV Task Overlay Cataloger (TOC) supports two new directives (SQU) and (CLE) and drops support of another (CHA). The SQUEEZE Directive allows the user to move one module at a time in order to consolidate as much free space together as possible. TOC no longer supports the CHANGE Directive. This function is now an OC and JM overlay. TOC has been modified to provide a command line extension to the TCB. The TOC processor now recognizes the CLE Directive and generates a TCB extension called Command Line Extension. This extension is 41 words long and the first word contains the length.

Directory manipulating directives lock out the load module file during the manipulation of the directory when TOCLOCK is enabled. If the TOCLOCK statement is NOT present in the System Structure Block (Part 1) of the SYSGEN or if TOCLOCK 0 is specified (in the System Structure Block) the lockout feature is NOT enabled.

Conversion Considerations

TOC, reuses freed disc space thus delaying the need for compressing. Segmented programs cataloged with versions of TOC prior to revision K.0 inaccurately indicate free space. To correct this the user must do one of the following before cataloging any new programs with revision K.0 or subsequent revisions of TOC:

- Delete all segmented programs and then recatalog them with revisions subsequent to revision K.0 of TOC.
- GET all programs and then PUT all programs under either revision of TOC.
- Compress the load module file under either revision of TOC. Complex duplicates cataloged with versions of TOC prior to Revision K.0, erroneously indicate increased sector usage. To obtain the correct size and free the space for cataloging other modules, recatalog each complex duplicate.

TOC32 REVISION A.0 SUMMARY

Revision A.0 of the MAX 32 Task Overlay Cataloger (TOC32) supports a unique directive called UNIX. This directive adds or deletes UNIX extensions in the TCB for cataloged tasks.

TABLE OF CONTENTS

Page

CHAPTER 1 OVERVIEW OF TOC/TOC32 1

1.1 TOC AND TOC32 1
1.2 TOC32 NOTES AND RESTRICTIONS 2

CHAPTER 2 SUMMARY OF DIRECTIVES 3

2.1 DIRECTORY HANDLING 3
2.2 FILE HANDLING 4
2.3 MODULE HANDLING 4
2.4 MODE SELECTION 4
2.5 INFORMATIVE 5
2.6 RESOURCE DEFINING 5

2.6.1 GENERAL 5
2.6.2 VIRTUAL SPACE 5

2.7 TASK CONTROL BLOCK 6
2.8 JOB CONTROL 7
2.9 MISCELLANEOUS 8

CHAPTER 3 DIRECTIVES 9

ACTION - Displays Message and Waits 10
ALLOCATE - Provides Memory for Task or Overlay 11
ASSIGN - Links Logical File to Logical File or Device 13
AVFILE - Advances File Forward Specified File-Marks 14
AVRECORD - Advances File Forward Specified Physical Records 15
BIAS - Positions Program at Location Other Than Virtual Address 0 16
BKFILE - Moves File Backward Specified File-Mark Records 18
BKRECORD - Moves File Backward Specified Records 19
BLOCK - Supplies New Data Chain Blocking Factors 20
CATALOG - Catalogs Task or Overlay On Selected Load Module 22
CLE - Reserves Addressing Space for Command Line 24
COMPRESS - Condenses Load Module File 25
COPY - Copies BI File to BO File 27
CTASSIGNMENTS - Clears Temporary Assignments 28
DEALLOCATE - Deallocates Memory 29
DEFAULT - Resets Resource Values to Default Values 31
DELETE - Removes Cataloged Task or Overlay From Current File 33
DUPLICATE - Gives Task or Overlay Additional Name 34
ERASE - Erases Memory Regions 36
EXIT - Leaves TOC or TOC32 38
FILE - Selects Load Module 39
GAP - Optimizes Load Time Efficiency 40
GET - Retrieve Task or Overlay 42

Page

HELP - Lists Mnemonics 43
ID - Catalogs Identification With Next Region 44
IIIALLOCATE Defines Maximum MAX III Allocations 45
INITIALIZE Clears and Initializes Directory 46
INSERT - Places Memory Regions into Unallocated Space 48
IOPERATIONS Queues I/O Operations Concurrently 53
ISPACE - Provides Extra Memory 54
KEY - Unlocks the Protected Load Module File 57
LAE - Lists Module Names Matching Mask 58
LDIRECTORY Reports Progress on Directory (No Parameters) 59
LIO - Reports Sector Usage of Current Load Module 60
LIST - Reports Programs in Current Load Module Directory 61
LOGFILE - Defines Logical File and Device Assignments 62
MAP - Shows Structure of Cataloged Program 64
MODIFY - Changes Instruction or Address 65
NOTE - Displays Message to System Operator 66
NOVERIFY - Verification Messages Are Not Printed Out 67
OPTIONS - Modifies Resource Information 68
OSPACE - Allocates Memory in Operand Space 70
OVERLAY - Sets Overlay Mode 71
PAGESHARE Inserts Page Sharing Regions in TCB 72
PAUSE - Displays a Message and Waits 73
PECULIAR - Modifies Resource Table for Unusual Task or Overlay 74
POPTIONS - Modifies Resource Information 78
PUT - Recatalogs a Program Retrieved by GET 80
REASSIGN - Defines or Redefines Logical Files 82
REMOVE - Deletes a Cataloged Task or Overlay 84
RENAME - Changes the Name of Cataloged Task or Overlay 85
RESTORE - Reconstructs Current Load Module 86
REWIND - Positions Device(s) to Beginning-Of-Medium 87
SAVE - Copies Entire Load Module File Onto BO File 88
SOURCE - Lists Source of Resources for Program 89
SPACE - Provides Extra Memory 91
SQUEEZE - Consolidates Free Space 94
STACKS - Defines Size of LIFO Stacks 95
SUBSTITUTE Changes Security Lock/Limit 97
SYSPAGES - Specifies Maximum Pages to be Allocated 99
TASK - Declares Next Program to be a Task 100
TEST - Checks Integrity of BI File 102
TIMELIMIT Limits Execution Time 103
TMTCB - Defines Two Maps 105
UNIX - Designates a UNIX Extension 106
USER - Adds User Extension to Next Task 107
VACATE - Inserts Vacant Entry 108
VARIABLE - Defines Variable Values 109
WEOF - Writes End-Of-File Mark 111
WORKBENCH Generates Workbench for Next Task 112

Page

CHAPTER 4 GUIDE TO THE USE OF TOC/TOC32 113

4.1 LOGICAL FILES 113
4.2 TYPES OF LOAD MODULES 114
4.3 INITIALIZING A LOAD-MODULE FILE 114
4.4 CATALOGING AN OVERLAY 115
4.5 CATALOGING A TASK 115
4.6 ARCHIVING A LOAD-MODULE FILE 116
4.7 RESTORING A LOAD-MODULE FILE 116
4.8 RECATALOGING A LOAD MODULE 117
4.9 RECATALOGING A TASK WITH NEW RESOURCES 117
4.10 RECATALOGING AND CHANGING THE TCB 118
4.11 JOB CONTROL PROCEDURES 118
4.12 CREATING LARGER CONTIGUOUS FREE SPACE 120

APPENDIX A QUICK LOAD FORMAT 121
APPENDIX B SECURITY FEATURES 123
APPENDIX C EXAMPLES OF CATALOGED PROGRAMS 125
APPENDIX D FORMAT OF MAP OUTPUT 127

D.1 SAMPLE TOC MAP OUTPUT 127
D.2 SAMPLE TOC32 MAP OUTPUT WITH DU OPTION 127
D.3 COLUMN HEADINGS 129
D.4 SELECTIONS FOR MAP OUTPUT 129

APPENDIX E SAMPLE TOC LISTINGS 137

E.1 SAMPLE LDI DISPLAY 137
E.2 SAMPLE LIS DISPLAY 138
E.3 SAMPLE LAE DISPLAY 139
E.4 SAMPLE LIO DISPLAY 140

APPENDIX F ERROR MESSAGES 143

GLOSSARY 157

INDEX 171

LIST OF FIGURES

1-1 Program Source Conversion Process 1

3-1 Memory Location of Unbiased and Biased Programs 17
3-2 Schematic of CATALOG Directive 22
3-3 Schematic of COMPRESS Directive 26
3-4 Schematic of DUPLICATE Directive 35
3-5 Inserting Global Shared Page Regions into a Program's Unused Addressing Space 49
3-6 Effects of ISPACE Directive on Memory Bias 56

Page

3-7 Schematic of PUT Directive with Identical Resources 80
3-8 Schematic of PUT Directive with New Resources 80
3-9 Schematic of SOURCE Directive 89
3-10 Effects of SPACE Directive on Memory Bias 93

4-1 Relationship of TOC/TOC32's Logical Files 113

LIST OF TABLES

3-1 Task's System Options 68
3-2 Task's Program Options 78

CHAPTER 1
OVERVIEW OF TOC/TOC32

1.1 TOC AND TOC32

Link editing is normally a prerequisite to cataloging. A successful link-edit operation combines object programs for a main program with object programs for all relevant subprograms and outputs the result in a single object program that has no unresolved references.

Such a self-contained object program can be added to a directoried disc file with TOC/TOC32. This operation is referred to as cataloging. The cataloged module is known as a load module, and the disc file as a load-module file. The load module can then be executed by the MAX IV/MAX 32 Module Loader.

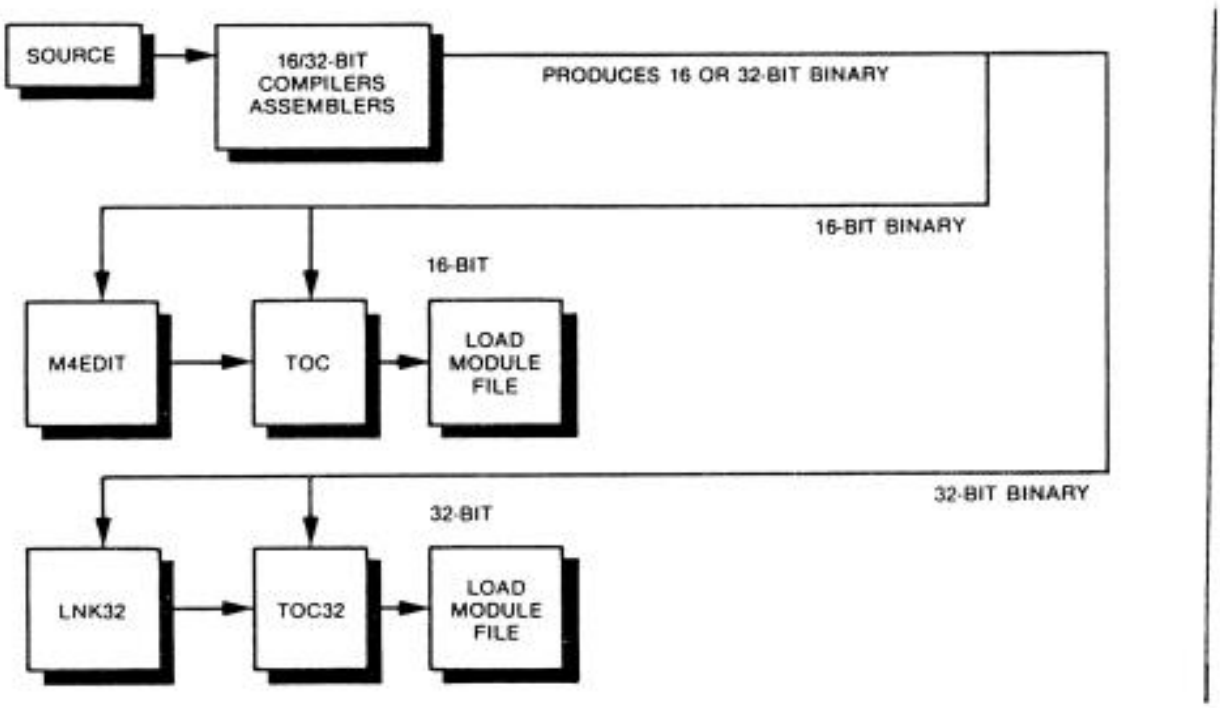


Figure 1-1. Program Source Conversion Process

TOC/TOC32 catalogs object programs output by the EDI or the M4EDI Link Editor as well as object programs that do not require link editing and are the output from any of the MODCOMP assemblers.

NOTE: TOC runs in a MAX IV or MAX 32 environment, and task overlay cataloger output can only be loaded by these operating systems. The 16-bit TOC programs can run as unprivileged programs under the MAX 32 Operating System. The 16-bit programs must be catalogued with TOC on separate load module files from those used by TOC32.

MAX IV/MAX 32 programs have a number of associated characteristics and resources that dictate the environment in which the loaded program operates. TOC/TOC32 transparently allocates default values to a user not requiring any special resources.

However, other users who require specific characteristics for a particular program use TOC/TOC32 to specify default values explicitly. The resources are cataloged together with the associated object program on a load-module file.

1.2 TOC32 NOTES AND RESTRICTIONS

The majority of TOC commands are valid for TOC32. Individual command differences are listed under that command.

The following commands are not supported:

- References to MAX III type resources.
- ALL parameter in memory allocation (Example: SPACE ALL,ALL is not valid). For more information, see the specific instructions.
- IMAP/OMAP in TOC32 are treated as errors.
- Two map options are treated as errors.

TOC32 relocatable programs are restricted to #80000 words. TOC32 is not required to be privileged to create a privileged load module.

CHAPTER 2
SUMMARY OF DIRECTIVES

This chapter summarizes the directives available to a user of TOC or TOC32. The directives can be categorized as follows:

- Directory Handling
- File Handling
- Module Handling
- Mode Selection
- Informative
- Resource Defining
- Task Control Block
- Job Control
- Miscellaneous

In order to understand the brief description accompanying each directive, the reader needs to know that TOC/TOC32 performs its input and output by addressing operations to logical files. In particular, during cataloging TOC/TOC32 reads input from the BI file writes its output through the BO file (with GET and COPY; CATALOG uses whatever file is specified) and outputs its listings to the LO file. A detailed description of logical files is given in Chapter 4. The descriptions of directives require some familiarity with MODCOMP terminology (Example: BI file, BO file, and LO file). (Refer to the Glossary.)

2.1 DIRECTORY HANDLING

These directives modify the directory:

CATALOG	Reads an object program from the BI file and catalogs it as either a task or overlay on the selected load-module file, using all the accumulated or default resource information that has been previously selected.
DELETE	Removes a program from the selected load-module file.
INITIALIZE	Initializes (clears) the directory of the selected load-module file so that cataloging can begin. (Destroys all existing information and load modules.)
REMOVE	Eliminates a program from the selected load-module file.
RENAME	Renames a program in the directory of the selected load-module file.
VACATE	Inserts a vacant entry at a specific point of the directory of the selected load-module file.

2.2 FILE HANDLING

These directives manipulate the files:

COMPRESS	Saves an entire load-module file to a specified scratch file, tests the copy and then restores it using only contiguous disc space. A listing of the directory is produced.
RESTORE	Restores a saved load-module file in its entirety, using only contiguous disc space as it is being restored. The original file or some new load module file can be the target. Reads from the BI file.
SAVE	Copies an entire load-module file, with directory, onto the BO file in a special save format. Vacant directory entries and deleted load modules are not copied.
SQUEEZE	Moves one load module at a time in order to consolidate free space.
SUBSTITUTE	Changes the security lock, file influence limit, and other file control options.
TEST	Reads and verifies a saved load-module file as written by the SAVE Directive. Reads from the BI file.

2.3 MODULE HANDLING

These directives manipulate the modules:

DUPLICATE	Duplicates an existing program in the selected load-module file's directory and assigns it a new, unique name. Optionally generates new, unique resources for the duplicate.
GET	Copies (in a special format) onto the BO file a load module cataloged on the selected load-module file.
PUT	Recatalogs a program originally retrieved by a GET Directive. Optionally substitutes new resource information in place of the original, leaving the object itself unmodified.

2.4 MODE SELECTION

FILE	Selects the logical file (must be assigned to a disc load-module file) on which the next program is cataloged.
KEY	Specifies the security key of a load-module file to which access is required.
NOV[ERIFY]	Prevents TOC/TOC32 from entering the hold state.
OVERLAY	Declares that the next program to be cataloged is an overlay of a task.

TASK	Declares that the next program to be cataloged is a task.

2.5 INFORMATIVE

These directives provide the operator with information:

LAE	Generates a report on LO of all module names that match the module name mask specified. The integrity of the modules is verified.
LDIRECTORY	Generates a report on LO of all items in the directory, but does not verify any module for integrity.
LIO	Generates a report on the LO file of the sector utilization in the current load module file.
LIST	Tests the selected load-module file for integrity of format and generates a report on the LO file containing a list of all cataloged programs and a summary of their characteristics.
MAP	Generates a report on the LO file describing all LPRs for a task.
SOURCE	Generates a report of the resources to LO. The directive can also copy the source of the resources to a file or initialize TOC/TOC32's internal tables with the resources.

2.6 RESOURCE DEFINING

These directives do not cause any activity when they are entered. The program characteristics set by these directives become effective during a cataloging activity.

2.6.1 GENERAL

DEFAULT	Causes all the resource values to be reset to default values.
ID	Causes a specific character string to be cataloged with the next program. The default is the current system date.

2.6.2 VIRTUAL SPACE

ALLOCATE	Allocates selected private memory regions of the next cataloged program's addressing space before the program is loaded.
BIAS	Relocates the next cataloged (non-link edited) program in its assigned virtual addressing space at some address other than the default (zero). TOC32 does not support BIAS.
BLOCK	Causes default-minimum data chain length and dummy read size to be overridden.

DEALLOCATE	Deallocates all or selected private memory regions of the next cataloged program's addressing space before it is loaded.
ERASE	Erases all or selected shared memory regions of the next cataloged program's addressing space before it is loaded.
GAP	Causes areas to be loaded with zeros in programs that do not have to be initialized when the program is loaded, which increases load-time efficiency (eliminates data chains).
INSERT	Causes shared load modules, global regions, private regions and copiable modules to be inserted into the next cataloged program's addressing space at load time.
ISPACE	(TOC only) Allocates additional virtual memory addressing the instruction space.
OSPACE	(TOC only) Allocates additional virtual memory addressing in the operand space.
SPACE	Allocates more virtual-addressing instruction space than is evident from its natural size and relocation bias. Optionally, it allocates additional private memory spaces to all or part of this additional-instruction space before the program is loaded.

2.7 TASK CONTROL BLOCK

CLE	Reserves space for the command line extension.
CTASSIGN	Clears all temporary assignments made by the LOGFILE or REASSIGN statements.
IIIALLOCATE	(TOC only) Defines the maximum number of MAX III memory allocations that can be active at any one time. The default is no MAX III memory allocation.
IOPERATIONS	Defines the maximum number of concurrent I/O operations the next cataloged task can queue up at any one time.
LOGFILE	Defines one or more logical files in the TCB of the next cataloged task and optionally binds device assignments to these logical files.
OPTIONS	Defines the states of one or more system option flags. The system can examine these when the program executes.
PAGESHARE	Defines the maximum number of page-sharing regions the next cataloged task can insert into its addressing space at any one time.
PECULIAR	Defines unusual traits to be granted by the operating system for the next cataloged program at time of execution.
POPTIONS	Defines the states of one or more program option flags. The program can examine these when it executes.

REASSIGN	Allows overlays to change the device assignments of one or more logical files of the task that loads it.
STACKS	(TOC only) Defines the initial size of two Last-In-First-Out (LIFO) stacks of the cataloged task. These stacks determine the task's ability to nest executive services and are extended dynamically as usage dictates.
SYSPAGES	Sets the maximum number of system pages that can be allocated.
TIMELIMIT	Defines the maximum CPU time the next cataloged task can consume during execution. The default is infinite time.
TMTCB	(TOC only) Declares that a two-map Task Control Block (TCB) is to be created for the next task to be cataloged.
UNIX	(TOC32 only) Designates a UNIX extension.
USER	Adds a user defined number of words to the task EXT.
VARIABLE	Defines one or more named or unnamed communication variables and optionally specifies initial values to these variables (one can be unnamed).
WORKBENCH	Reserves space for the workbench extension.

2.8 JOB CONTROL

ACTION	Puts the task in hold informs the user of a procedure that must be performed before cataloging can continue.
ASSIGN	Causes one or more logical files of the host batch processing task to be assigned or to default.
AVFILE	Positions file/device after forward file mark.
AVRECORD	Skips file forward one or more record.
BKFILE	Positions file/device ahead of previous file mark.
BKRECORD	Skips file backward one or more record.
NOTE	Causes the user to be informed without suspending operations.
REWIND	Positions a file/device at beginning-of-medium.
WEOF	Writes an end-of-file mark onto a selected logical file.
$NOP	Performs no operation. Inserts filler records by Job Control procedures.

2.9 MISCELLANEOUS

COPY	Copies a file from the BI file to the BO file.
EXIT or E	Exits TOC/TOC32, returns to Job Control.
HELP	Provides a list of TOC/TOC32's mnemonics.
MODIFY	Causes the next program to be patched before it is cataloged.
PAUSE	(Same as ACTION.)

CHAPTER 3
DIRECTIVES

Directive statements must be entered in the specific format required by TOC or TOC32.

The following symbols and style are used in Chapter 3 to convey this required format for entering directives.

EXAMPLES

REW[IND]	file	[file]
OPT[IONS]	optioname REMO[VE]	[optioname]

[ ] - Brackets represent optional. Brackets around part of a word indicate insignificant characters; brackets around an entire word indicate an optional item (Example: directive or parameter). For each TOC/TOC32 directive, only the first three characters are significant.

upper-case - Words in upper-case indicate alphanumeric characters to be entered by the operator.

lower-case - A descriptive word in lower-case represents a variable to be replaced with a value by the operator. If one word is not a sufficient description, concatenation is used (linked words).

n - The letter 'n' is a variable for a value.

... - An elipsis indicates a continuing sequence.

...n - An elipsis followed by the letter 'n' indicates an indefinite sequence.

arabic number - An arabic numeral is the identifier of the parameter (for the purpose of discussion in the text).

ACTION

Display a Message and Wait for Response

The ACTION Directive (same as PAUSE Directive) outputs a message on the CO and LO file and waits for operator response. TOC/TOC32 is placed into the hold state.

Enter either a RESUME (/R) Directive to resume execution or an ABORT (/A) Directive to abort.

SYNTAX

ACT[ION] [text]

[text] - Parameter 1 (optional) displays a message directed to the operator's attention. If not entered, the message 'ACTION' is displayed.

EXAMPLES

ACTION SET THE WRITE-PROTECT SWITCHES FOR DISC
The message 'ACTION SET THE WRITE-PROTECT SWITCHES FOR DISC' is output to the CO file.

ACT SET THE WRITE-PROTECT SWITCHES FOR DISC
The message 'ACT SET THE WRITE-PROTECT SWITCHES FOR DISC' is output to the CO file.

Allocate Pages of Memory when the Task or Overlay is Loaded

The ALLOCATE Directive causes private memory to be allocated to the task or overlay. The task or overlay aborts when it is loaded if the task's map is not large enough to hold the allocated pages (refer to the SPACE Directive, Parameter 2). This directive can prevent inefficient use of memory space. For instance, when loading a 72-page task with 51 pages of allocated space, first 72 pages and then 51 pages are allocated. If 51 pages are not available, the task waits (with 72 pages already allocated and not available to any other task). A better approach is to do an ALLOCATE of 123 pages (51 plus 72). This way, if 123 pages are not available, no memory is tied up.

SYNTAX

ALL[OCATE] [IMAP] / [OMAP] [$]startpage numpages / REMO[VE]

[IMAP]
[OMAP] - Parameter 1 (optional for TOC; must be **omitted** for TOC32) can be either IMAP or OMAP. It specifies the map into which memory is allocated. (Default = IMAP) IMAP specifies Instruction Map. OMAP specifies Operand Map.

[$]startpage - Parameter 2 (required) specifies the starting page to be allocated. The optional dollar sign ($) specifies that the load bias is added to the starting page.

numpages
REMO[VE] - Parameter 3 (required) must be either numpages or REMOVE.

'numpages' specifies the number of pages to be allocated.

REMOVE specifies that the allocate is to be removed from TOC/TOC32's internal tables. This is not processed until TOC/TOC32 encounters a CAT or PUT,,NEW Directive.

TOC EXAMPLES

ALLOCATE IMAP,0,123	Allocate pages 0 through 122 to IMAP.
ALLOCATE 0 123	Allocate pages 0 through 122 to IMAP.
ALL IMAP,$17,1	Allocate the page at BIAS/256 +17 to IMAP.
ALL 38,REMO	Remove from the IMAP the pages previously specified by an ALLOCATE #38 statement.
ALL OMAP,#14,12	Allocate pages #14 through #20 in the OMAP.

TOC32 EXAMPLES

ALLOCATE 0,123	Allocate pages 0 through 122 in map.
ALL 38,REMOVE	Remove a previously specified ALLOCATE 38.

Link a Logical File to a Logical File or to a Device

The ASSIGN Directive links logical file(s) to physical device(s) or to other logical file(s). A logical file assigned to itself (for example, ASS LO LO) defaults, if there is no default it becomes vacant. The ASSIGN Directive can be used to remove a logical file with a default assignment from the File Assignment Table (FAT). If 'ASS LO =,' is entered the entry for LO in the FAT becomes a vacant entry. The load module file last specified in a FILE Directive cannot be made vacant.

A new file of a task can be opened by ASSIGN if that task has vacant file assignments in its File Assign Table (FAT). The name specified in Parameter 1, if not already in the FAT, is opened if a vacancy exists. Such new files are not given a default assignment.

If the logical file specified in Parameter 1 is assigned to a device, then the File Position Index in the FAT is reset to 0. If it is assigned to another file, then its File Position Index becomes that of the file specified in Parameter 2.

Refer to the MAX IV or MAX 32 BASIC I/O SYSTEM System Guide Manual for further information about logical files.

SYNTAX

ASS[IGN] logfileone logfiletwo
devicename

logfileone - Parameter 1 (required) specifies the logical file of the task to be bound to a device or other logical file.

logfiletwo
devicename - Parameter 2 (required) must be either logfiletwo or devicename.

'logfiletwo' specifies the name of another logical file already assigned and can be a global file name.

'devicename' specifies the name of a system device to be linked to the logical file being assigned.

EXAMPLES

ASSIGN BI=BO	Assign binary input to binary output.
ASS BI MT1	Assign binary input to magnetic tape drive 1.

AVFILE

Advance the Logical File Forward x File-marks

The Advance File (AVFILE) Directive moves a specified logical file forward. The File Position Index (FPI) in the File Assign Table (FAT) for the specified logical file is changed. The FPI is incremented by 1 for each physical record encountered until the number of file marks specified in Parameter 2 is passed.

For a description of the response of a particular device to the AVFILE Directive, refer to the appropriate MAX IV/MAX 32 handler manual listed in the Preface.

SYNTAX

AVF[ILE] filename [integer]

filename - Parameter 1 (required) specifies the name of the logical file to be advanced.

[integer] - Parameter 2 (optional) specifies the number of file-mark records to be advanced. If it is not specified, a value of 1 is used.

EXAMPLES

AVFILE B0 Advance logical file B0 until one file-mark record is passed.

AVF B1,2 Advance logical file B1 until two file-mark records are passed.

Advance the Logical File Forward Past Physical Records

The Advance Record (AVRECORD) Directive moves the specified logical file forward past the number of physical records specified in Parameter 2. The File Position Index (FPI) in the File Assign Table (FAT) is incremented by 1 each time a physical record is encountered. For a description of the response of a particular device to the AVRECORD Directive, refer to the appropriate MAX IV/MAX 32 handler manual listed in the Preface.

SYNTAX

AVR[ECORD] filename [integer]

filename - Parameter 1 (required) specifies the name of the logical file to be advanced.

[integer] - Parameter 2 (optional) specifies the number of records to be advanced. If it is not specified, a value of 1 is used.

EXAMPLES

AVRECORD BO Advance logical file BO one record.

AVR BI,2 Advance logical file BI two records.

NOTE: If the specified file reaches its End-of-Medium position before the count is satisfied, the following message is displayed on the CO file (which is usually assigned to the terminal):

END OF MEDIA

The specified logical file remains positioned beyond the last record until another file positioning directive is processed.

BIAS (TOC only)

Position the Program at a Fixed Position in its Addressing Space Other than Virtual Address 0

The BIAS Directive (TOC only) loads an overlay or task at an address other than 0 in the addressing space.

By default, a program being cataloged in the quick-load format is fixed in the addressing space at relative address 0. This directive relocates the program at cataloging time at some other address. The addressing space between relative address 0 and the start of the biased program is unused virtual space (on integral page boundaries) and is not initially allocated actual memory when the program is loaded. This space can be utilized by the executing program for dynamic memory allocation or for inserting page-sharing regions.

NOTE: Do not use the BIAS Directive with programs that are link edited by the MAX IV Link Editor (M4EDIT).

The BIAS Directive is used only when cataloging programs. BIAS is ignored by the directive statement PUT,,NEW.

SYNTAX

BIA[S] value

value - Parameter 1 (required) specifies a decimal or hexadecimal word address to be added to all relocatable addresses. The default value is 0. The value must be in the range of 0 to 64K (0-#FFFF).

EXAMPLES

BIAS #AB00 Add the hexadecimal value of #AB00 to all relocatable addresses.

BIA #4000 Refer to Figure 3-1 on the next page.

NOTE: All unallocated virtual-space regions are given access rights of 00 (non-existent memory).

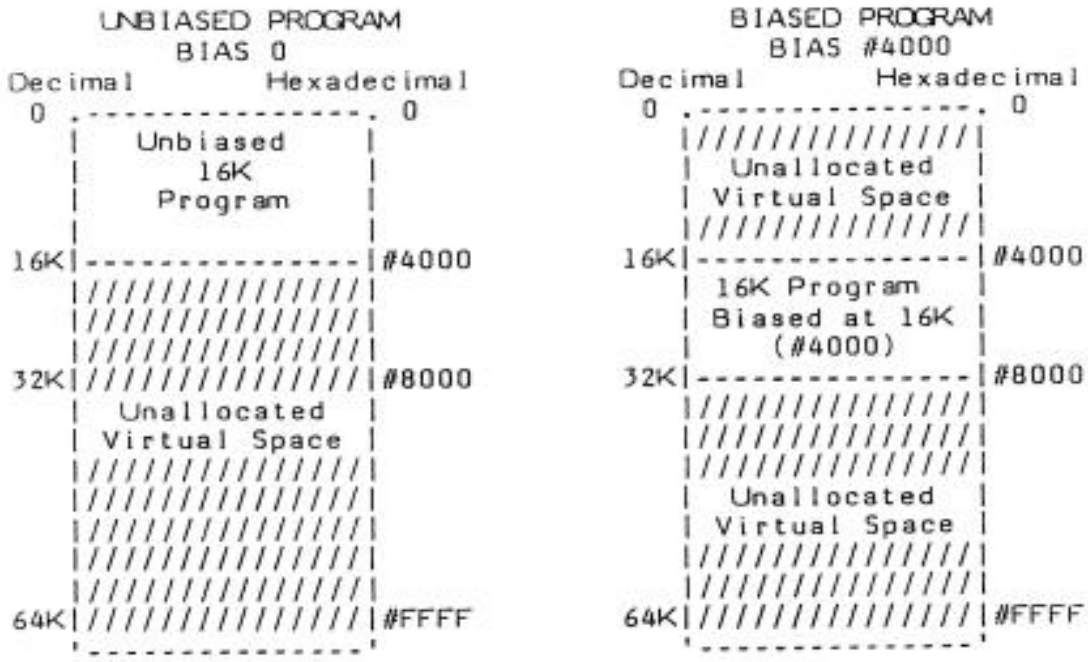


Figure 3-1. Memory Location of Unbiased and Biased Programs

BKFILE

Move the Logical File Backward x File-mark Records

The Back File (BKFILE) Directive moves a device medium in the reverse direction until the number of file-mark records detected is equal to the number specified in Parameter 2. Once the beginning-of-medium position is reached, TOC/TOC32 ignores any amount remaining in Parameter 2.

The File Position Index in the File Assign Table is decremented by 1 for each physical record encountered.

For a description of the response of a particular device to the BKFILE Directive, refer to the appropriate MAX IV/MAX 32 handler manual listed in the Preface.

SYNTAX

BKF[ILE] filename [integer]

filename - Parameter 1 (required) specifies the name of the logical file to be moved backward.

[integer] - Parameter 2 (optional) specifies the number of file-mark records to be moved backward. If it is not specified, a value of 1 is used.

EXAMPLES

BKFILE BI Move logical file BI backward one file-mark record.

BKF BI,2 Move logical file BI backward two file-mark records.

Move the Logical File Backward x Records

The Back Record (BKRECORD) Directive moves the device medium in the reverse direction until the number of physical records skipped is equal to the number specified in Parameter 2. If the particular device cannot perform such an operation, then no operation is performed.

Once the beginning-of-medium is reached, TOC/TOC32 ignores any amount remaining in Parameter 2.

For a description of the response of a particular device to the BKRECORD Directive, refer to the appropriate MAX IV/MAX 32 handler manual listed in the Preface.

SYNTAX

BKR[ECORD] filename [integer]

filename - Parameter 1 (required) specifies the name of the logical file to be moved backward.

[integer] - Parameter 2 (optional) specifies the number of records to be moved backwards. If it is not specified, a value of 1 is assumed.

EXAMPLES

BKRECORD BI Move logical file BI backward one record.

BKR BI,2 Move logical file BI backward two records.

BLOCK

Supply New Data Chain Blocking Factors

The BLOCK Directive causes default minimum data chain length and dummy read size to be overridden. BLOCK is reset to default values whenever a FILE, COMPRESS, RESTORE, or ASSIGN (of the attentive file) is processed.

The third parameter, which is reset only upon entering a FILE Directive, is active for all functions that catalog entries onto a load-module file (CATALOG, PUT, and RESTORE). It is reset to its default during a COMPRESS operation and then set to its previously entered value. Refer to the GAP and SPACE Directives for more information about the use of this directive.

Users with controllers that run under the LX.HAN (4176A, 4180, etc.) do not need to consider the BLOCK command.

SYNTAX

BLO[CK] [min] [dummy] [max]

[min] - Parameter 1 (optional) specifies the minimum data chain length in words. Default is:

Controller Model	Words
4125/4136	2
4100	10
All other models	1

[dummy] - Parameter 2 (optional) specifies the number of words read by a dummy read. Default is:

Controller Model	Words
4125/4136	2
4100	12
All other models	1

[max] - Parameter 3 (optional) specifies the maximum size in words of any one data chain. Default is 16384 words.

EXAMPLES

BLOCK 8	Causes the current minimum data-chain-block length (the minimum length a Transfer Count (TC) must be before a data chain is not broken) to be set to 8.
BLOCK,15	Supply a dummy read size of 15 words.
BLOCK,,15	Supply a dummy read size of 15 words.
BLOCK,,#1000	Limit the size of any one data chain TC (Read Module Object (RMO) chain list entry) to #1000 words.
BLOCK,,,#1000	Limit the size of any one data chain TC (RMO chain list entry) to #1000 words.

Catalog the Task or Overlay Described on the Load Module File

The CATALOG Directive reads an object program from the BI file and catalogs it either as a task or overlay on the selected load-module file, using all the accumulated or default resource information that has been previously selected.

The program is given an alias name and its basic potential memory resources, and if it is a task, it is assigned a priority, influence limit, and additional resources. Other directives define these resources. A schematic of the actions taking place during the execution of the CATALOG Directive is displayed in Figure 3-2.

If this directive is preceded by the two-map Task Control Block (TMTCB) Directive, then a two-map TCB is created although the task might have been link-edited with one-map (TOC only). TOC uses the largest space available to catalog this module.

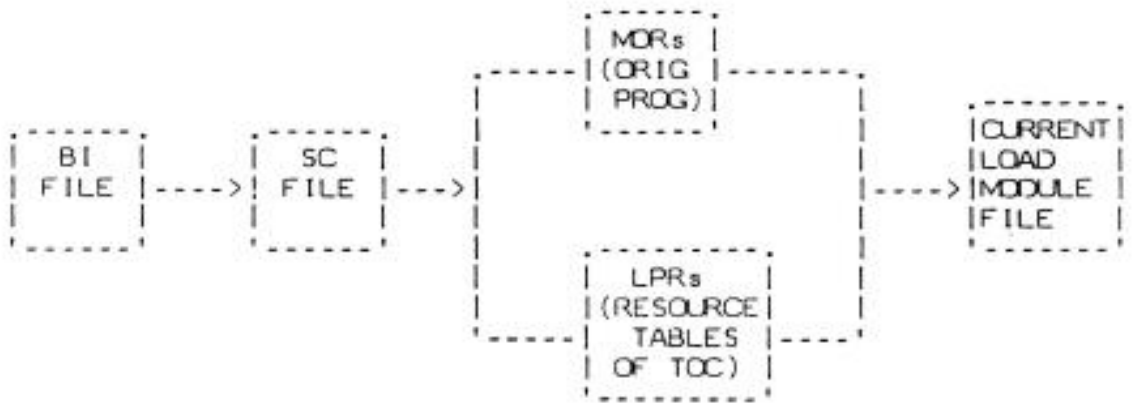


Figure 3-2. Schematic of CATALOG Directive

When DUPLICATE or INSERT DUPLICATE is specified, the CATALOG Directive does not read from the BI file and the program name is ignored.

SYNTAX

CAT[ALOG] [pgmname]

[pgmname] - Parameter 1 (optional) specifies the program name, specified by the Program (PGM) Directive, of the pre-edited object program residing on sequential file BI to be reformatted into the Quick-Load format. If the name is missing or is MAIN, the next program encountered on the sequential BI file is cataloged.

EXAMPLES

CATALOG MYPROG	Search for MYPROG on BI and catalog it if found.
CAT MAIN	Catalog next program found on sequential file BI.
CAT	Catalog next program found on sequential file BI.
TMTCB CAT	Two-map task (TOC only). Catalog next program found on logical file BI and if it is a task, then create a two-map TCB for it.

CLE

Add Room for the Command Line Extension to the Next Task Cataloged or PUT

CAUTION

This directive is only compatible with Revision 1.0 (or later) of the MAX IV Operating System or with the MAX 32 Revision A.0 (or later) Operating System.

The CLE Directive causes the command line extension to be generated for the next task cataloged. CLE is an optional feature of the MAX IV/MAX 32 Operating System that requires extra memory space. The purpose of specifying the CLE Directive is to reserve addressing space for the command line feature of the Operating System.

The command line area is pointed to by EXTCLE in the task TCB extension when that task is active.

SYNTAX

CLE [REMO[VE]]

[REMO[VE]] - Parameter 1 (optional) indicates that the command line extension is to be removed for the next task.

EXAMPLES

CLE Reserve addressing space for the command line extension.

CLE REMO Remove the command line extension.

Condense the Load Module File

The COMPRESS Directive can be entered at anytime. Its purpose is to consolidate any possible fragmentation of the directory or program storage space of the current load-module file. The COMPRESS Directive consolidates and saves an entire load-module file with directory to a specified scratch file, verifies the copy, and then copies it back using only contiguous disc space. The COMPRESS Directive performs save, test, restore, and list functions.

CAUTION

The COMPRESS Directive does not operate if there are duplicate masters that have been removed, unless either:

- The duplicate master is recataloged;
  or
- The duplicate is also removed.

When the test function is complete, a message of 'SAVE COMPLETED /task name/TOC' is output to the CO file. If a power failure or some other system failure occurs after the SAVE COMPLETED message, restore the file manually (refer to RESTORE Directive) when the system comes back up. If a power failure occurs before this message, the original file is not changed and can be compressed again.

After the restore function is complete, the load module directory is output to the CO file. Refer to the LIST Directive for additional information.

If a load-module file has a task connected to a task-scheduling timer, the load-module file cannot be compressed and the LPR pointers are not reliable. Since the processing of this directive embodies a save and restore request (function), an unprivileged TOC/TOC32 cannot compress a file containing privileged programs.

Refer to Figure 3-3 on the next page for additional information on the use of this directive.

TOC/TOC32 reuses the space after a module has been deleted, so the need for a compress has been delayed until the load module file is entirely fragmented.

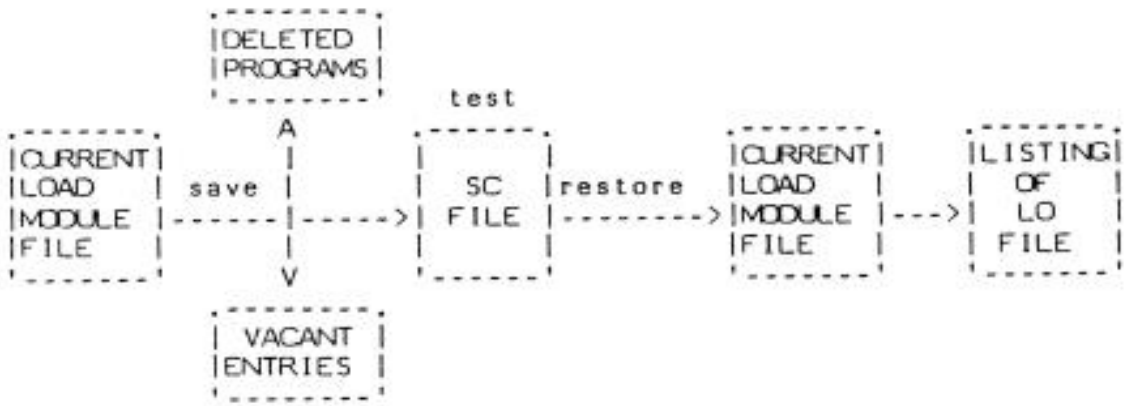


Figure 3-3. Schematic of COMPRESS Directive

SYNTAX

COM[PRESS] file

file - Parameter 1 (required) specifies the logical name of a scratch file and must be large enough to hold the entire load-module file being compressed in the save format.

EXAMPLES

COMPRESS SC Compress load-module file using scratch partition SC.

COM SC Compress load-module file using scratch partition SC.

Copy the BI File to the BO File

The COPY Directive reads standard binary records from the BI file, reblock these records, and outputs them to the BO file with a new checksum, binary ID (03 or 07), and sequence number (if necessary) until an end-of-file (EOF) is encountered on BI. COPY writes an EOF on BO when an EOF is read on BI. The sequential BI and BO files must be manually positioned.

Use GET, PUT, SAVE and RESTORE to move directoried load module files.

The COPY Directive accepts any standard object file including:

- Assembler binary output.
- Library Update binary output.
- MAX III or IV Link Editor binary output.
- LNK32 output.

The output of the COPY Directive is checked for data integrity by the TEST Directive. Records from BI are read and examined. Data integrity (checksum) and record sequence are verified. The records are reblocked to the physical-device size or 128 words, whichever is less, and a new checksum and sequence number are generated if necessary.

SYNTAX

COP[Y]

EXAMPLES

COPY	Copy a file from the BI file to the BO file.
COP	Copy a file from the BI file to the BO file.

CTASSIGNMENTS

Clear All Temporary Assignments to Make Room for Loading the Next Overlay

The Clear Temporary Assignments (CTASSIGNMENTS) Directive clear all temporary assignments made by LOGFILE or REASSIGN statements when the next cataloged program is loaded, making vacancies available in the File Assign Table (FAT).

For information on the use of these vacancies, refer to the REASSIGN Directive.

Without this directive invoked, the host task must be cataloged with enough vacant logical files for this and subsequent overlays. The default is that no temporary assignments are cleared.

SYNTAX

CTA[SSIGNMENTS] [RE[MOVE]]

[RE[MOVE]] - Parameter 1 (optional) specifies that temporary assignments are not to be cleared. (It removes the CTASSIGNMENTS flag.)

EXAMPLES

CTASSIGNMENTS	Clear all temporary assignments made by LOGFILE or REASSIGN statements.
CTA RE	No temporary assignments made by LOGFILE or REASSIGN statements are to be cleared.

Deallocate Pages of Memory when the Next Overlay is Loaded

The DEALLOCATE Directive deallocates all or selected private memory regions of the next cataloged program's addressing space before it is loaded. DEALLOCATE does not need to be specified for tasks because new memory resources are constructed for tasks when they are loaded. This directive is used for an overlay to ensure that only enough memory is left in the root task to contain the new chain overlay.

NOTE: Only use this directive for overlays that are to receive control.

If the root task program is ever reloaded explicitly as an overlay, use this directive to maintain its appropriate memory size (for example, cataloging the Job Control overlay of a batch processing task with the DEALLOCATE ALL statement saves memory space).

SYNTAX

```
DEA[LLOCATE]   [IMAP]   ALL              [REMO[VE]]
               [OMAP]   [$]startpage     numpages
                        REMO[VE]
```

[IMAP]
[OMAP] - Parameter 1 (optional for TOC; must be omitted for TOC32) can be either IMAP or OMAP. It specifies the map from which memory is deallocated. If neither parameter is specified, IMAP is used.

IMAP specifies Instruction Map.

OMAP specifies Operand Map.

ALL
[$]startpage
REMO[VE] - Parameter 2 (required) must be either ALL, [$]startpage, or REMOVE.

ALL specifies that all memory is to be deallocated before loading of the module begins. Space for this module is reallocated later in the loading process.

'[$]startpage' specifies the starting page to be deallocated. The optional dollar sign ($) specifies that the load bias is added to the starting page.

REMOVE specifies that the deallocate is to be removed from TOC/TOC32's internal tables. TOC/TOC32 processes a CAT or a PUT,,NEW Directive as if the DEALLOCATE startpage or the DEALLOCATE ALL had never been entered.

[REM[OVE]] numpages - If ALL is specified for Parameter 2, Parameter 3 (optional) can be REMOVE.

REMOVE eliminates the DEALLOCATE ALL.

If [$]startpage is specified for Parameter 2, Parameter 3 (required) must be numpages.

'numpages' specifies the number of pages to be deallocated.

EXAMPLES

DEALLOCATE ALL	Deallocate all memory in addressing space from IMAP.
DEA 0,17	Deallocate first 17 pages from IMAP.
DEA $4,5	Deallocate pages BIAS/256 + 4 for 5 pages from IMAP.
DEA ALL,REMO	Remove the DEALLOCATE ALL from IMAP.
DEA 4,REMO	Remove the deallocate of pages 4 through 8 from IMAP.
DEA OMAP,0,3	Deallocate 3 OMAP pages (TOC only).

Reset All Resource Values to Default Values

The DEFAULT Directive causes all the resource values to be reset to default values. When TOC/TOC32 is initially loaded, its tables are initialized to default values. After cataloging a program, these tables can have been selectively modified by directives. Use this directive to clear tables to their default values. The default parameters for all programs are as follows:

- Alias name of the next cataloged program: natural six-character name.
- Priority and influence limit: unspecified (actual task priority is determined at execution time).
- Task push stack size: 100 words and 12 words, and two I/O nodes are provided. The main stack is extensible (TOC only).
- File assignments: none, and an overlay does not change file assignments.
- Variables: none, and loading an overlay does not change variables.
- Page-sharing: none.
- Tasks' System Options (OPTIONS) and Program Options (POPTIONS): reset to 0, and loading an overlay does not change OPTIONS or POPTIONS.
- Relocation bias: 0 (program's addressing space(s)).
- No additional virtual space is added to a task's map image(s).
- No additional private memory, except for natural program size, is allocated. No shared regions are edited into the next program's addressing space(s).
- Peculiarities: none.
- ID: system date at the time TOC/TOC32 is executed.
- System Pages (SYSPAGES): 2.
- User area of the EXT: 0.

SYNTAX

DEF[AULT]

EXAMPLES

DEFAULT	Reset all resource values to default values.
DEF	Reset all resource values to default values.

Remove a Cataloged Task or Overlay from the Current File

The DELETE Directive removes the specified task or overlay program from the directory of the selected load-module file. DELETE can be executed at any time.

The directive executes a different process depending on whether the program is a duplicate or an original:

- Duplicate - removes program from directory.
- Original - removes program from directory and all duplicates are recovered during the next COMPRESS operation to the file.

The REMOVE Directive performs this same function.

The space used by the deleted module is reused when a new program is cataloged and this space is part of the largest space available.

SYNTAX

DEL[ETE] name

name - Parameter 1 (required) specifies the alias name that was cataloged with the task.

EXAMPLES

DELETE FRANK Remove program FRANK from the directory.

DEL BRENT Remove program BRENT from the directory.

DUPLICATE

Give a Cataloged Task or Overlay an Additional Alias Name

The DUPLICATE Directive causes a program, already cataloged, to be made accessible under more than one alias name. This is useful for tasks, since only one task can be active under a single name. TOC/TOC32 does not duplicate an existing duplicates.

If the duplicate master is deleted, delete all the duplicates before doing a COMPRESS or SAVE operation.

Refer to Figure 3-5 for additional information.

SYNTAX

DUP[LICATE] masternname dupname [NEW[RESOURCES]]

masternname - Parameter 1 (required) specifies the alias name of a program, task, or overlay already assumed to be cataloged on the load-module file. This is called a duplicate master.

dupname - Parameter 2 (required) specifies the new alias name to be entered into the file directory and then linked to the duplicate master.

[NEW[RESOURCES]] - Parameter 3 (optional) determines the resources duplicates use and what occurs when a duplicate master is recataloged:

. Parameter 3 specified (complex duplicates)

- Programs share a single program body but use unique resources.
- Complex duplicates are elimiated unless also recataloged.

. Parameter 3 not specified (non-complex duplicates) - Programs share same resources.

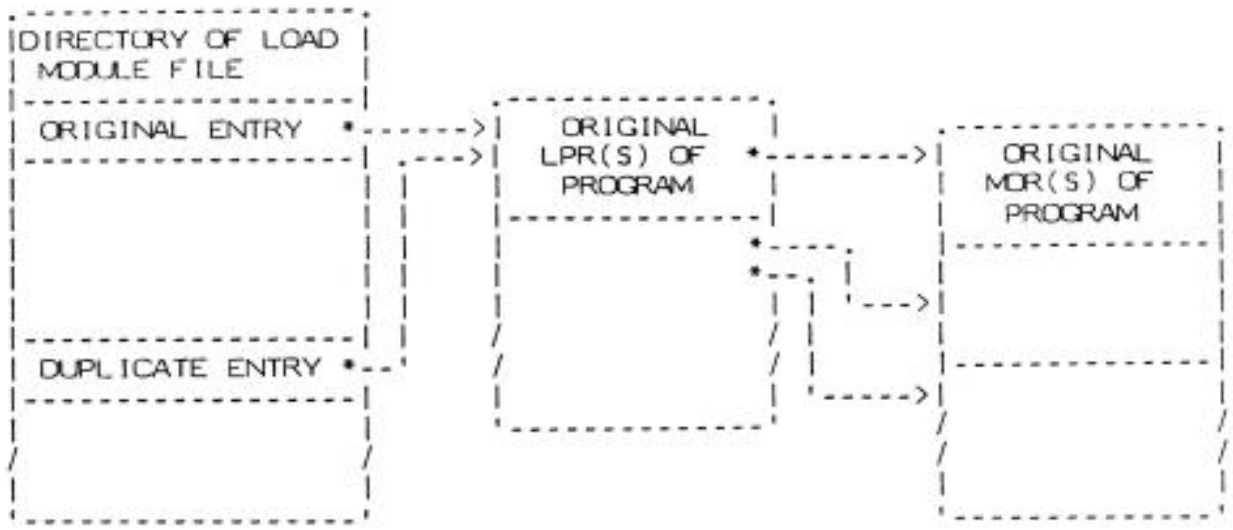
- Non-complex duplicates are reattached to the new copy.

EXAMPLES

DUPLICATE JOHN,TEDDY,NEWRESOURCES
Create TEDDY as a duplicate of JOHN with different resources.

DUP RICHARD,TOM
Create TOM as a duplicate of RICHARD with identical resources.

DUPLICATE (SIMPLE):



DUPLICATE WITH NEWRESOURCES:

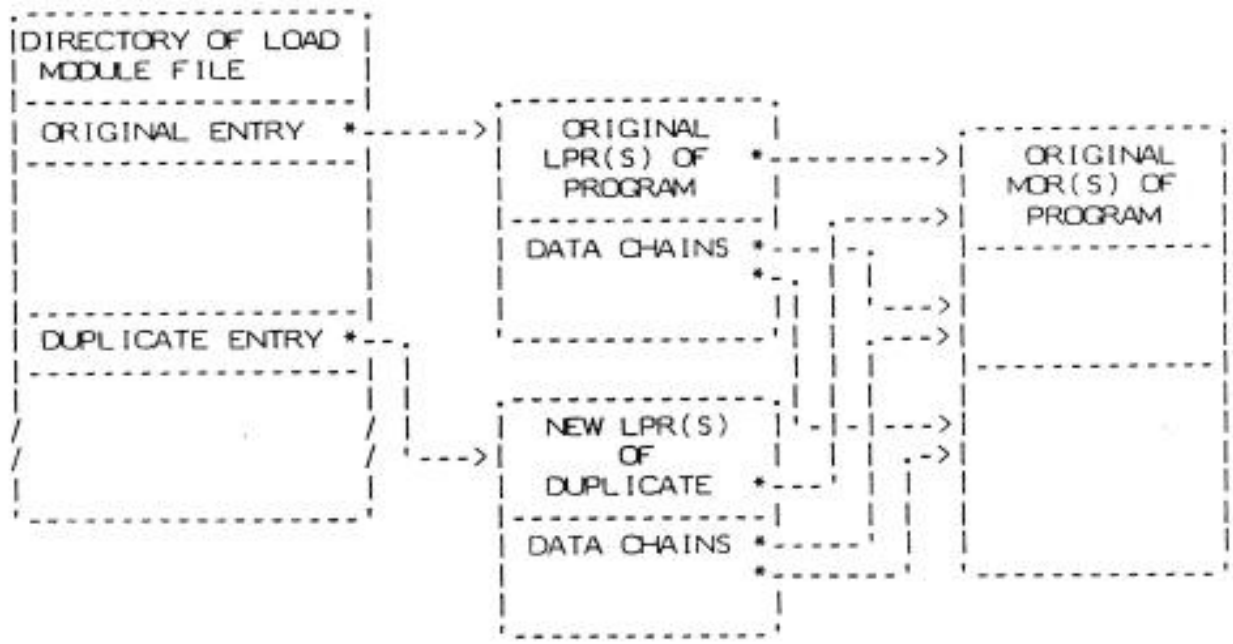


Figure 3-4. Schematic of DUPLICATE Directive

ERASE

Erase Shared Memory Regions When Loading the Next Overlay

The ERASE Directive erases all or selected shared memory regions of the next cataloged program's addressing space before it is loaded. This directive does not need to be specified for tasks because totally new memory resources are constructed for all root tasks when they are activated. For an overlay to use shared memory regions that do not exist in the root task, the overlay must specify this directive in the range of addressing space where new shared memory is to be invoked. This directive must be used with all overlays that insert shared pages where there might be preexisting shared pages.

If the root task uses the RELOAD ROOT option for exit or abort, or if the task has been overlayed by a new overlay, this directive may be required. The Job Control program, which is cataloged as a batch processing task, uses the ERASE ALL statement to ensure that all shared-memory regions are erased whenever it is reloaded.

SYNTAX

```
ERA[SE]    [IMAP]   ALL         [REMO[VE]]
           [OMAP]   startpage    numpages
                    REMO[VE]
```

[IMAP]
[OMAP] - Parameter 1 (optional for TOC; must be omitted for TOC32) specifies the map from which the shared memory regions are erased and can be either IMAP or OMAP. (Default = IMAP)

IMAP specifies Instruction Map.

OMAP specifies Operand Map.

ALL
startpage
REMO[VE] - Parameter 2 (required) must be either ALL, startpage, or REMOVE.

ALL specifies that all shared memory regions are to be erased in the root task's addressing space before loading of the module has begun. This module can be loaded in the space where shared pages existed for the root task or a previous overlay.

'startpage' specifies the starting shared page where the erasure begins.

REMOVE specifies that the erase is to be removed from TOC/TOC32's internal tables.

[REMO[VE]] numpages - Parameter 3 is required only if startpage is selected as Parameter 2.

'numpages' represents the number of shared pages to erase.

REMOVE specifies that the erase is to be removed from TOC/TOC32's internal tables.

EXAMPLES

ERASE ALL	Erase all shared regions from IMAP.
ERA 0,20	Erase first 20 pages from IMAP.
ERA 6,4	Erase pages 6 through 9 from IMAP.
ERA ALL,REMO	Remove the ERASE ALL from IMAP.
ERA OMAP 0,3	Erase shared regions from OMAP in pages 0 through 2 (TOC only).

EXIT

Leave TOC/TOC32

The EXIT Directive enables the user to stop using TOC/TOC32 and return to Job Control.

SYNTAX

E[XI[T]]

EXAMPLES

EXIT	Exit from TOC/TOC32 and return to Job Control.
EXI	Exit from TOC/TOC32 and return to Job Control.
E	Exit from TOC/TOC32 and return to Job Control.

This is the Logical File to Catalog On

The FILE Directive selects the load-module file (must be assigned to a disc load module file) that is processed by the next action directive (for example, to catalog a program or compress a file). If the file is locked, a security key must be specified prior to entering this directive (refer to KEY Directive).

Access is not granted if TOC/TOC32 is executed at a batch-processing level with a limited influence limit and the file's directory has a higher influence limit.

The load-module file can be changed as often as necessary.

NOTE: Disc partitions must have unlimited record size 0 and a geometry of 1.

SYNTAX

FIL[E] logfilname

logfilname - Parameter 1 (required) specifies a 1- to 3-character logical file name. This logical file must be assigned to a disc device suitable for quick-load formatted data. Suitable disc partitions have unlimited record size 0 and a geometry of 1.

EXAMPLES

FILE BM	Select batch-processing-load-module file.
FIL LM	Select foreground-task-load-module file.
FIL SM	Select system-load-module file.

Optimize Load Time Efficiency

The GAP Directive causes areas in programs (that need not be initialized when the program is loaded) to be filled with zeros until either the minimum Transfer Count (TC) is attained or another Transfer Address (TA) or Transfer Count is found. This increases load-time efficiency (eliminates data chains).

Data chaining is a problem when small TCs are used with a non-fully-buffered disc. Data chaining requires three-memory accesses plus one-memory access per data word to be stored. A one-word transfer requires a memory access to:

- Pick up the TA
- Pick up the TC
- Store the data word
- Store the updated TA

So the smaller the TC, the faster the disc.

The minimum TC for a DDC Fixed-Head Disc defaults to 10 words. The minimum TC for the ISS/AMPEX Disc defaults to 2 words. If consecutive transfers with TCs of less than 10 words are used, the module loader tries to give data from the disc faster than memory can accept it. If this directive is not used and if a TC of less than 10 words is found, TOC/TOC32 breaks the data chain (a new RMO), wastes the rest of the sector, and can lose a disc revolution (lost load time) before the next read is initiated.

To prevent data loss, absolute TA/TCs and TA/TCs that load into space lower than the previously loaded address space are not affected by GAP.

The user is referred to the BLOCK, SPACE, ISPACE, and OSPACE Directives for additional information on data chaining.

Users with controllers that run with the LX.HAN (Example: 4176, 4180) should not need to worry about this command.

SYNTAX

GAP [min] [max]

[min] - Parameter 1 (optional) specifies the minimum amount of memory in words (GAP) between data chain elements. (This is the area not normally loaded where one TA+TC ends and the next TC starts). In memory, zeros are output by TOC/TOC32 starting at the first TA and ending at the second TA-1 (if equal to or less than min), eliminating the second TA/TC and conserving LPR space.

[max] - Parameter 2 (optional) specifies the maximum amount of memory between data chain elements to fill.

EXAMPLES

GAP — Connect TA/TCs that are 1 or 2 words apart. Also connect or fill all TA/TCs that are more than 2 words apart if the last TC is so small that it would break a data chain (new RMO).

GAP 12 — This is the same as the first example except that all space between TA/TCs of 12 or less words are connected.

GAP 3,9 — This is the same as the first example except that all space between TA/TCs of 3 or less words are connected and a maximum of 9 words of fill are supplied even if it breaks a data chain.

GET

Retrieve a Task or Overlay in the Current Load-Module File and Put it in a Special Object Module Format

The GET Directive causes a program already cataloged on the load-module file to be transferred to a sequential file in a format that permits it to be recataloged later as an individual program on this or some other load-module file (refer to PUT Directive).

All directory information, LPRs, and MORs are reformatted so that they are written sequentially on the BO file in a special save format. This save format can be only be read by the PUT Command. A file mark is not automatically written on BO at the end of a GET. This directive cannot operate on a privileged module if the currently executing TOC/TOC32 is unprivileged. Duplicates can not be retrieved with a GET Directive.

Refer to the COPY Directive for information about moving the GET output to other devices.

SYNTAX

GET name

name - Parameter 1 (required) specifies the name of a program already cataloged on the currently selected load-module file.

EXAMPLES

GET TOC	Retrieve the MAX IV/MAX 32 Task Overlay Cataloger program.
GET B	Retrieve the batch task B.

List the Mnemonics

The HELP Directive provides a listing on the LO file of all directive mnemonics, peculiar types, insert types, system options, and program options. In addition the directive mnemonics are broken down into those supported by TOC/TOC32 and by its overlay OVTOC.

SYNTAX

HEL[P]

EXAMPLES

HELP	Help me with the TOC/TOC32 mnemonics.
HEL	Help me with the TOC/TOC32 mnemonics.

ID

Catalog this Identification With the Next Program

The ID Directive causes the specified text string or the current date to be written in the LPRs of the next and subsequent cataloged tasks or overlays.

This information is ignored by the system loader but is included in the written report generated by the LIST Directive when the cataloged program appears in that report. If the ID Directive is not entered or does not include the text parameter, the current date is obtained from the system's Time-of-Day clock, converted to a text string, and used as the indentifier in the format:

mm/dd/yy

SYNTAX

ID [text]

[text] - Parameter 1 (optional) is a string of a maximum of eight characters of identification. (Default = a character string generated from the current system date maintained by the Time-of-Day clock.)

EXAMPLES

ID LEE6.74	The identifier is 'LEE6.74'
ID	Use current date as ID.

Allow MAX III Memory Allocations in the Next Task Cataloged

The MAX III Allocate (IIIALLOCATE) Directive (TOC only; not supported under TOC32) defines the maximum number of MAX III allocations that can be active at any one time. It allows the next task cataloged to perform MAX III memory allocation/deallocation REX calls. The default is no MAX III memory allocation.

SYNTAX

III[ALLOCATE] max

max - Parameter 1 (required) is the maximum number of MAX-III-type memory-allocation calls that can be active at any one time (cannot be greater than 253).

EXAMPLES

III 1	Only allow one memory allocation or deallocation at a time.
III 5	Only allow five active memory allocations or deallocations at a time.

INITIALIZE

Clear the Current File and Give it an Empty Load-module Directory

The INITIALIZE Directive clears the directory of the selected load-module file so that cataloging can begin. It eliminates any information existing in the file and intializes it with an empty load-module directory.

CAUTION

Use either the SAVE or GET Directive prior to entering the INITIALIZE Directive to prevent data loss.

Use the INITIALIZE Directive to start cataloging programs on a new load-module file when no directory structures currently exist for the disc device assigned to that file or to destroy all existing programs cataloged on an existing load-module file (to begin cataloging new programs).

To prevent unintentional data loss, this directive requires operator intervention unless the directory is not initialized or has been initialized with NOVERIFY.

Only disc files with geometry 1 and unlimited record size can be initialized.

Partition size should not exceed 32768 sectors.

If using the File Manager, ensure that the load-module file is completely contiguous.

SYNTAX

INI[TIALIZE]	[level]	[security]	[NOV[ERIFY]]	[directory sector]	[duplicate directory sector]

[level] - Parameter 1 (optional) is the minimum limit priority of the cataloger that can access the file to be initialized. Default is 255.

[security] - Parameter 2 (optional) is the security key (CAN-code) that must be entered to the cataloger by the KEY Directive before the cataloger can access this file. KEY is initialized to security by this Directive and KEY does not have to be entered as long as the cataloger remains in memory accessing the file that was initialized. If no security key was entered, the security key is not checked by the FILE, LIST, or ASSIGN Directives.

[NOV[ERIFY]] - Parameter 3 (optional) specifies that no verification of this file is to be done. When recataloging a program that is already in the directory, without this mode, the cataloger halts to allow the operator to verify the cataloging of this new module. With this mode, and the NOVERIFY Directive previously specified, cataloging takes place without the operator being notified. The message 'NOVERIFY WANTED' is displayed for operator verification.

directory sector - Parameter 4 (optional) specifies the number of sectors of the load module file that are set aside for the directory (defaults to one).

duplicate directory sector - Parameter 5 (optional) specifies the number of sectors of the load module file that are set aside for the duplicate directory (defaults to one).

Since directory sectors are generated as needed, Parameters 4 and 5 are useful in setting up the number of directory sectors needed before any cataloging is done. The SQUEEZE function does not move directory sectors.

Directory sectors allocated by the INITIALIZE Directive that are actually unused are deleted if a COMPRESS is done. Compress places directory sectors at the beginning of the file while additional duplicate directory sectors are placed after the last load module.

EXAMPLES

INITIALIZE	Destroy all programs on current file.
INI 253,AL,NOV	Set the minimum cataloger limit priority as 253; set security key of AL; set No Verification Mode.
INI,,,NOVERIFY	Set Noverify Mode with initialization.

INSERT

Insert a Memory Region into Unallocated Space

The INSERT Directive causes shared-load modules, global regions, private regions, and copyable modules to be inserted into unallocated virtual-space regions of the next cataloged program.

More than one such region can be inserted into a program if the program has sufficient virtual space and if a sufficient number of entries have been specified for the PAGESHARE extension of the host task. Extended memory regions do not require a PAGESHARE extension.

Insert private or global regions into object programs output from any compiler (except a MAX IV Assembler) during link editing (before TOC is running).

If the next program being cataloged is an overlay, the virtual region into which a shared memory region is being inserted must be cleared of all allocated memory and preexisting shared memory regions (refer to DEALLOCATE and ERASE Directives). A task that has no preexisting memory or virtual space need not be cleared unless it is also an overlay.

If WRITE is specified as Parameter 6, it is possible for a program to load all or parts of itself into shared regions since the insertion of shared regions is accomplished before the cataloged program is loaded. Insert Copyable Module (ICM) causes the module loader to be called repeatedly.

(Refer to the MAX IV or MAX 32 GENERAL OPERATING SYSTEM, System Guide Manuals.)

Both the INSERT COPY and INSERT DUPLICATE statements use the Module Loader's ICM. The INSERT COPY statement inhibits change of the loader extension. The loader extension contains the high, low, and start addresses of the module. The INSERT DUPLICATE statement allows the loader extension to be updated by the nested call to the module loader.

The INSERT DUPLICATE statement performs the same function as the DUPLICATE Directive. The body of a task can be cataloged on a load-module file as an overlay without resources. The tasks themselves could be cataloged with resources and the overlay containing the task body inserted at load time. The major advantage of this approach over complex duplicates is that if the body is updated or recataloged the tasks inserting that body are not affected. Another advantage is a decrease in the amount of disc space needed to catalog the task.

After specifying the INSERT DUPLICATE Directive the CATALOG Directive puts the new task on the load-module file. Refer to Figure 3-5.

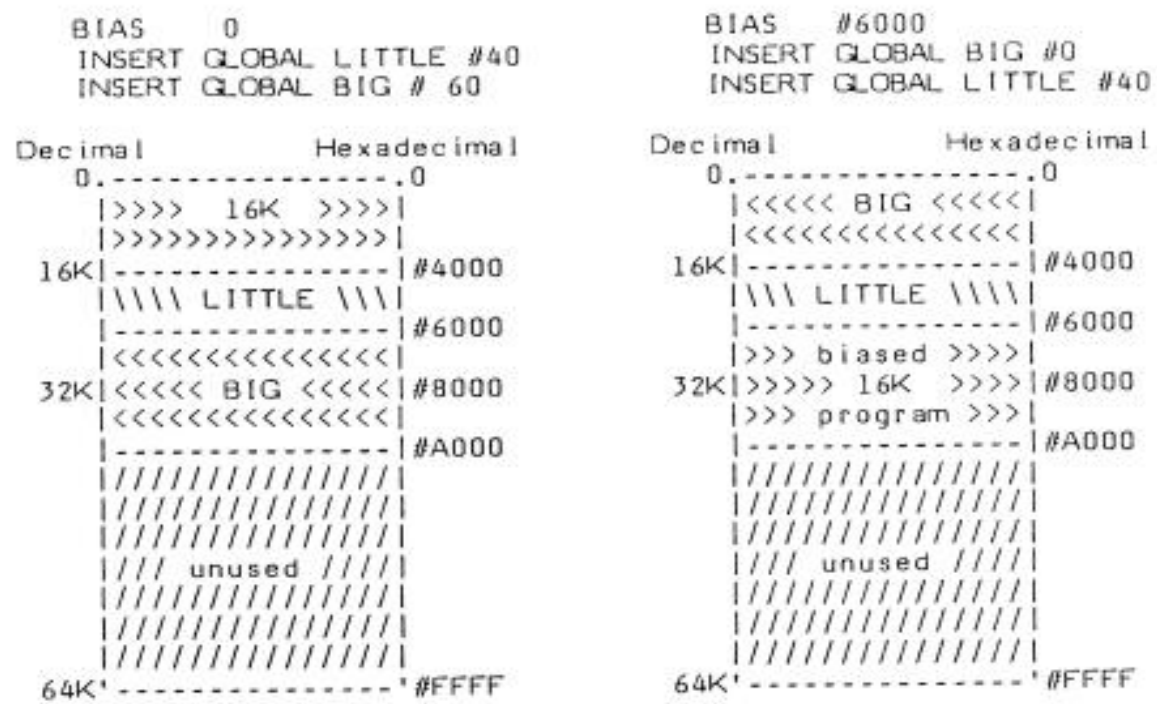


Figure 3-5. Inserting Global Shared Page Regions into a Program's Unused Addressing Space

Figure 3-5 illustrates two basic ways of inserting global-shared page regions into a program's unused addressing space. The first example inserts the shared regions at pages above the program. The second example biases the cataloged program so that the shared regions can be inserted below the program. If the BIAS step is skipped, the program loads itself into the inserted global regions.

SYNTAX

INS[ERT]

[IMAP]	PRI[VATE] name	[page]	[readkey]	[writekey]	W[RITE]
[OMAP]	GLO[BAL] name	[REMO[VE]]			R[EAD]
	SLM name,lfn				E[XECUTE]
	COPY name,lfn				
	DUP name,lfn				

[IMAP] - Parameter 1 (optional for TOC; must be omitted for TOC32) can be either IMAP or [OMAP] OMAP. It specifies the map into which the region is inserted. If neither parameter is specified, IMAP is used.

IMAP specifies Instruction Map.
OMAP specifies Operand Map.

PRI[VATE]name - Parameter 2 (required) specifies the type of GLO[BAL] name region to be inserted into the next cataloged SLM name,lfn program's addressing space and must be either COPY name,lfn 'PRIVATE name', 'GLOBAL name', 'SLM DUP name,lfn name, lfn', 'COPY name,lfn', or 'DUP name, lfn'.

PRIVATE regions are allocated from the memory pool when first requested and exist only as long as at least one task is still using them.

GLOBAL regions are groups of contiguous actual pages permanently defined at system generation.

SLM specifies a shared-load module.

COPY specifies a copiable module.

DUP specifies a duplicate copy.

'name' is required for Parameter 2 and it is the 1- to 6-character name of the desired shared memory region to be inserted. It follows the keyword and is separated from the keyword by a space.

'lfn' is required for Parameter 2 if SLM, COPY, or DUP is used for Parameter 2. It follows 'name' and it is separated from 'name' with a comma. It specifies the logical file name from which the module loads. If this parameter is absent TOC/TOC32 uses the last logical file name loaded. This parameter is not present for GLOBAL or PRIVATE inserts.

[page]
[REMO[VE]] - Parameter 3 (optional) can be either page or REMOVE.

'page' is the virtual page number in the program's addressing space where the shared memory region is inserted.

If no page number is specified, page 0 is used. This and all pages following it must be sufficient to hold the named region and must be unallocated and have no previously inserted regions. If this is an insert on an Extended Memory Region, the value does not affect the insert.

If the page number is preceded by a dollar sign ($) it indicates that when the overlay or task is loaded the inserted region is to be loaded at address "page" relative to the inserting module. If the dollar sign is not present then "page" will be treated as an absolute address.

REMOVE specifies that the INSERT of name is to be removed from TOC/TOC32's internal tables. Inserting a region with the same name as a previous insert does not delete the previous insert. INSERT must first be removed before an insert of the same name is entered.

[readkey] - Parameter 4 (optional) is a 1- to 3-character security key that needs to be specified only if the named region being inserted has a read lock specified when defined.

[writekey] - Parameter 5 (optional) is a 1- to 3-character security key that need be specified only if the named region being inserted has a write lock specified when defined and WRITE access is requested in Parameter 6.

W[RITE] - Parameter 6 (required) specifies the desired R[EAD] memory access right to be given all the pages E[XECUTE] of the inserted region and must be either WRITE, READ, or EXECUTE. The default is WRITE. Refer to the MAX IV or MAX 32 GENERAL OPERATING SYSTEM System Guide Manual for more information concerning memory accessing.

WRITE specifies write access.
READ specifies read access.
EXECUTE specifies execute access.

EXAMPLES

INSERT OMAP,PRIVATE,BRIAN,75,ACE,W

Insert the private region called BRIAN into the OMAP at page 75 with a security key of ACE. The region name BRIAN has a read lock specified. A write access is given to the region.

INS IMAP,SLM,JIM,LM,65,E

Insert the shared load module region called JIM into the IMAP. The module is loaded from the LM file and an execute access is given to the region.

INS OMAP,GLOBAL,PAM,REMO

Remove the previously defined global region called PAM from the OMAP.

The Task Queues Many I/O Operations Concurrently

The Input/Ouput Operations (IOPERATIONS) Directive sizes a multi-entry list structure in the EXT of the next cataloged task. Each entry is a data structure called an I/O node that is used as a queuing vehicle that manages an I/O operation through the system. Every task must have at least two nodes, therefore, two private nodes are automatically allocated when the task is loaded. The task program can queue multiple operations to different devices or to the same device (for example, double buffering). Excessive use of private I/O nodes is wasteful. Only a few programs are designed to sustain more than three or four concurrent I/O operations.

SYNTAX

IOP[ERATIONS] numprivate-1 [numglobal]

numprivate-1 - Parameter 1 (required) specifies one less than the maximum number of I/O operations that the next cataloged task is allowed to queue concurrently without using global I/O nodes. The default value is 1, but one additional private I/O node is allocated when the task is loaded so default is actually 1 plus 1.

[numglobal] - Parameter 2 (optional) specifies the maximum number of global I/O nodes the task can use. When the task runs out of private I/O nodes, the task can allocate up to 'numglobal' number of I/O nodes from the global pool for temporary use (if the global nodes are available). The default value is 0.

EXAMPLES

IOPERATIONS 3	Define four private I/O nodes.
IOP 1,4	Define 2 private nodes with a maximum of four global nodes.

ISPACE (TOC only)

Give the Next Task or Overlay Extra Memory Space in the Instruction Map

(TOC only.) The Instruction Space (ISPACE) Directive (same as SPACE Directive) makes the next cataloged task allocate more virtual addressing instruction space than is evident from its natural size and relocation bias. Optionally, it allocates additional private memory space to all or part of this additional instruction space before the program is loaded. If ISPACE or SPACE is not entered, the next task or overlay is loaded into memory space assumed to exist or be sufficient to hold the program body after any optional bias. ISPACE permits additional allocated or unallocated memory space to be bound to the program when it is loaded. This space is allocated above the natural program taking into account a bias (refer to Figure 3-6). Only a task can request additional virtual memory space. An overlay only allocates additional space within the virtual space limits already owned by its host task. A series of REX services can be used by a task or overlay to assign actual memory to unallocated space at execution time.

The Allocated Parameter is useful for data chaining when the transfer counts are too small on fast discs. (Refer to the description on the GAP Directive before reading further.) If GAP cannot be used or if backward or absolute transfer addresses are encountered, use the Allocated Parameter to increase load-time efficiency with a minimum of extra space being allocated.

If a one word TC is encountered while cataloging on one of these fast discs, a dummy read can be placed after the one word TC so that another TC of one word can follow. The Allocated Parameter must be specified so that TOC can give the Load Module Loader a buffer to read into. Allocating just one word causes an entire page (256 words) to be allocated.

Refer to the BLOCK Directive for additional information about overriding default values for data chaining.

SYNTAX

ISPA[CE] [allocated] [virtual]
[ALL] [ALL]

[allocated] - Parameter 1 (optional) can be either allocated or ALL.
[ALL]

'allocated' is the number of words of allocated memory to be given to the program in addition to the memory size allocated for its natural instruction space. The default value is 0.

ALL specifies that the rest of virtual instruction space is wanted no matter what size the virtual space is. This parameter is useful for programs like Sort/Merge that need all available memory to run efficiently.

[virtual] - Parameter 2 (optional) can be either virtual
[ALL] or ALL.

'virtual' is the number of words of unallocated virtual addressing space in addition to all natural and allocated instruction space. The default value is 0. Only tasks can use this parameter.

ALL specifies that the next cataloged task is able to have a full IMAP (up to 64K words of memory).

EXAMPLES

ISPAC 500,500	Add 500 words of space, 500 more words that can be allocated at run-time.
ISPACE #1000	Add 4096 words of allocated space.
ISPACE 0,#400	Allow up to 1024 more words to be allocated at run-time (by the task).
ISPACE ALL	Allocate maximum memory.
ISPACE 0,ALL	Allow up to 64K words of memory to be allocated at run-time (by the task).

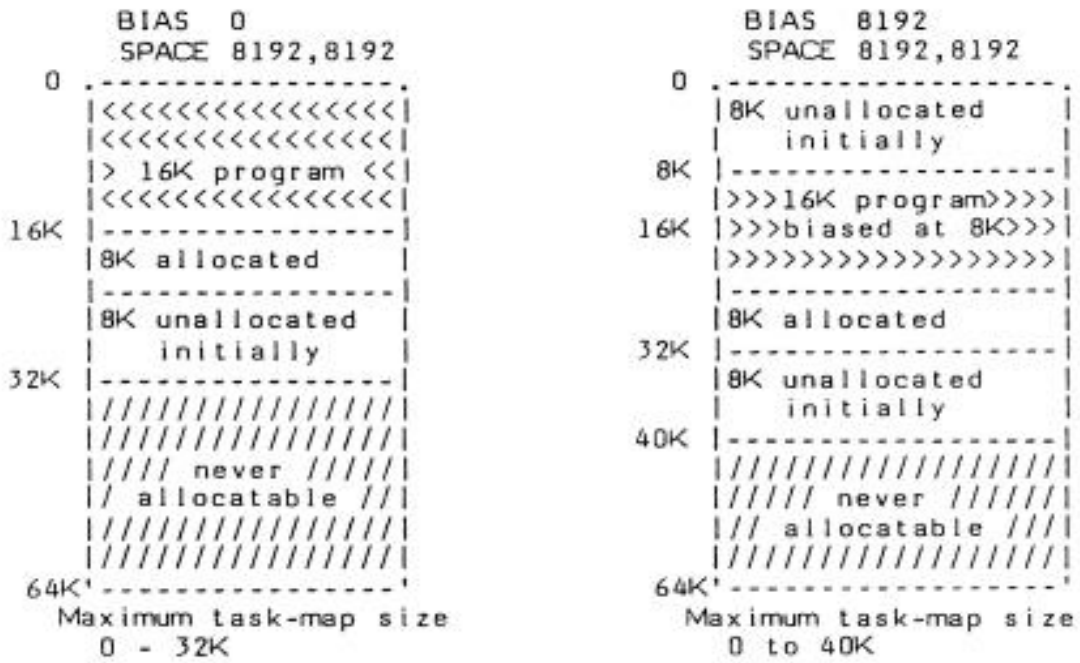


Figure 3-6. Effects of ISPACE Directive on Memory Bias

This is the Security Code that Unlocks the Load-Module File

The KEY Directive specifies the security key of a load-module file to which access is required. This directive is not required if the load-module file to be selected was not initialized with a security lock. If a locked file is referenced by a FILE, INITIALIZE, or ASSIGN Directive, the following message is printed on the CO file if the specified key does not fit the lock:

ILLEGAL SECURITY KEY /task/TOC

If the Control Input (CI) file is not assigned to an interactive terminal device, TOC/TOC32 is aborted with the reason code 'TOC'. This flushes the batch job from the job stream.

To disable the key entered on the last KEY Directive enter another KEY Directive without an argument.

SYNTAX

KEY [string]

[string] - Parameter 1 (optional) is a 1- to 3-character CAN-codable string that matches the lock specified by the most recent INITIALIZE or SUBSTITUTE Directive for the load-module file to be selected for cataloging.

EXAMPLES

KEY ABC Specify a security key of ABC.

KEY Remove last security key entered.

LAE

Report Selected Items in the Current Load-Module File's Directory

The List All Entry (LAE) Directive generates a report on logical file LO of the module names that match the specified mask. The programs are verified for integrity. This directive is similar to LIST except not all modules are listed (refer to Appendix E).

SYNTAX

LAE string

string - Parameter 1 (required) consists of one to six characters, each of which is either a valid character for a program name or a plus sign (+).

EXAMPLES

LAE +S+	Generate a report of all three character program names that have an "S" as the second character.
LAE T+++++	Generate a report of all module names that start with the letter "T".
LAE ++++++	All module names are reported and verified for integrity.

Report All Items in the Current Load-Module File's Directory

The List Directory (LDIRECTORY) Directive generates a report on logical file LO of all programs in the current load-module file's directory. The programs are not verified for integrity. This directive is similar to the LIST Directive, except not all the parameters are printed since programs are not loaded (refer to Appendix E).

SYNTAX

LDI[RECTORY]

EXAMPLES

LDIRECTORY	Generate a report of all programs in the current load-module file's directory.
LDI	Generate a report of all programs in the current load-module file's directory.

LIO

List in Sector Order the Usage of the Sectors in the Current Load Module

The List in Order (LIO) Directive reports the sector usage of a load module file. All sectors are identified as either unused, used in a directory sector, or used in a module. This directive identifies the actual disc area that is not used. The system manager can move modules through judicious use of the GET, DELETE, and PUT Directives to reduce disc fragmentation. Refer to Section 4.12 for an example of this usage. TOC/TOC32 catalogs new modules in the largest hole currently available. See Appendix E for example of output.

SYNTAX

LIO

EXAMPLES

LIO Generate a report of sector usage.

NOTE: Complex duplicates and segmented programs cataloged with versions of TOC prior to Revision K.0 will show an inaccurate number of sectors in the LIO listing. Refer to Appendix E.4.

Verify Integrity of the Load-Module File and Make a Report of All its Programs

The LIST Directive generates a report on logical file LO of all programs in the current load-module file's directory. The program prints one line of information for each program in the load-module file. Additionally, if Parameter 2 is not specified, all module's LPRs are loaded and verified for integrity (refer to Sample List Display in Appendix E).

SYNTAX

LIS[T] [logfilename] [DI[RECTORY]]

[logfilename] - Parameter 1 (optional) specifies the logical file name of a cataloged load-module file already containing a directory of cataloged programs. The default is the last load-module file specified by the FILE Directive.

[DI[RECTORY]] - Parameter 2 (optional) specifies that only the directory is to be listed; the modules are not loaded and verified for integrity. (This keyword performs the same function as LDIRECTORY).

EXAMPLES

LIST LM	Generate a report on logical file LO of all programs on load-module file LM and verify for integrity.
LIS LM,DI	Generate a report of the logical file LO of all programs on load-module file LM.
LIS	Generate a report on logical file LO of all programs on file specified by FILE Directive and verify for integrity.

LOGFILE

Define This Logical File and its Assignments for the Next Cataloged Task

The Logical File (LOGFILE) Directive defines one or more logical files in the TCB of the next cataloged task and optionally binds device assignments to these logical files. LOGFILE is repeated, as necessary, to define all the required logical files of the next task to be cataloged. A data structure in its TCB is sized according to the maximum number of logical files defined (vacant and with specific assignments). An overlay only adds logical files where vacancies exist or only changes the assignments of existing logical files (refer to REASSIGN Directive). A logical file can be assigned to another logical file as well as to a device. In this case, however, the assignee must have been previously defined.

SYNTAX

LOG[FILE]	logfilename number REMO[VE]	normal REMO[VE]	[default]	[TEM[PORARY]]

logfilename
number
REMO[VE] - Parameter 1 (required) must be either logfilename, number, or REMOVE.

'logfilename' is a 1- to 3-character CAN-code name of a logical file to be defined for the task.

'number' is the amount of logical file entries defined for the task without initial assignments. The maximum number of vacant files is 254.

REMOVE eliminates the previously defined vacant log. (If more than one vacant log definition is made, the first definition is used.)

normal
REMO[VE] - Parameter 2 (required) must be either: normal or REMOVE.

'normal' is a 1- to 3-character name of a physical device or logical file that is assigned to the logical file when the task is loaded. It can be the name of another logical file, if the other logical file is already defined and assigned.

REMOVE erases LOGFILE assignments of the logical file specified in Parameter 1 from TOC/TOC32's internal tables. This parameter is useful when a LOGFILE is to be reassigned or removed from the resources of a task (refer to SOURCE Directive) or if a typing error is made when entering the LOGFILE Directive. If the incorrect entry is not removed, it remains in TOC/TOC32's internal tables.

[default] - Parameter 3 (optional) states the 1- to 3-character name of a physical device or logical file that serves as a secondary assignment for the logical file. If Parameter 3 is not specified, it defaults to the device named in Parameter 2.

If a comma comma (,,) is input default remains unassigned.

[TEM[PORARY]] - Parameter 4 (optional) specifies that this LOGFILE is temporary and can be removed by the CTASSIGNMENTS Directive when another module is loaded.

EXAMPLES

LOGFILE LO,LP	LO to printer.
LOGFILE CI,CR	CI from card reader.
LOGFILE JC,AJC	Job Control procedures from disc file.
LOGFILE 5,TY	FORTRAN file to teletypewriter.
LOGFILE 6	Six vacant files.
LOG IN,TY2	Input file to teletypewriter.
LOG OUT,TY2	Output file to teletypewriter.
LOG ABC,LP,LPX	LP normal; LPX default.
LOG PQR,LP,,	LP normal; no default.
LOG PQR,REMO	Remove PQR.
LOG REMO	Remove previous vacant log.

NOTE: The sum of all LOGFILE statements and vacant entries cannot exceed 254.

MAP

Show the Structure of the Cataloged Program

The MAP Directive generates a report on the LO file describing all loader function codes in the same format displayed when loaded by MAX IV/MAX 32. The information includes how the function codes are contained within the LPRs and how the records are located on the disc.

Refer to Appendix D, which defines the map output format.

SYNTAX

MAP [name]

[name] - Parameter 1 (optional) is the name of a program already cataloged on the currently specified load-module file. (Default = All programs on the file have a map printed.)

EXAMPLES

MAP B	Report all loader function codes contained within the LPRs of B.
MAP	Report all loader function codes of every program on the active file.

Change An Instruction or Address in the Next Program Cataloged or PUT

The MODIFY Directive changes an instruction or address within the next program before TOC/TOC32 catalogs or puts it. The MODIFY Directive is applicable only for one-map tasks or overlays.

All MODIFY Directives are placed in a table with room for 50 modifies. Neither reserves (SAC op-code) nor space outside of the program can be modified. Any modifies not processed by TOC/TOC32 during a CATALOG or PUT Directive are listed on the LO and CO files, if CO is assigned to an input/output device. The Modify table is cleared to 0 after every CATALOG or PUT Directive.

All modifies are done on a 16-bit basis. TOC32 mode of 32-bit values must be done with two separate 16-bit values.

SYNTAX

MOD[IFY] [$]addr [$]valu0,[$]valu1,...n

[$]addr - Parameter 1 (required) specifies the address to be modified or, if more than one address is to be modified, it specifies the starting address. The dollar sign ($) specifies that the current BIAS is be added to 'addr'.

[$]valu0,
[$]valu1,
.
.
.
n

- Parameter 2 (required) and Parameters 3 through n (optional) specify a value for each address to be changed. For example, if #57 was specified for Parameter 1, the content of #57 changes to valu0 (Parameter 2); #58 changes to valu1 (Parameter 3); #59 changes to valu2 (Parameter 4); and so on. The dollar sign ($) specifies that the value is relative (load time offset is be added to 'valu') and the current bias is addded to the value.

EXAMPLES

MODIFY #57,#E700,$#28
Put #E700 in location #57 and put relative #28 in location #58.

MOD $#175 0 Add bias to #175 and put 0 in that location.

NOTE

Display a Message With No Response Required

The NOTE Directive sends a message to the user on the LO and CO files while a job stream is being executed. TOC/TOC32 is not placed into the HOLD state and no operator intervention is required.

Although only the first three characters (NOT) are required for the NOTE Directive it is recommended that all four characters (NOTE) be typed to avoid confusion. (Refer to second example below.)

SYNTAX

NOT[E] [text]

[text] - Parameter 1 (optional) contains the text of the message to be output to the CO file. The maximum number of characters for the text is governed by the device to which the CO file is currently assigned.

EXAMPLES

NOTE PROGRAMS ARE NOW BEING CATALOGED

The message 'NOTE PROGRAMS ARE NOW BEING CATALOGED' is output to the CO file.

NOT END OF JOB STREAM X

The message 'NOT END OF JOB STREAM X' is output to the CO file. Notice that dropping the directive's insignificant character 'E' results in a message that is the opposite of the intended message.

NOT

The message 'NOT' is output to the CO file.

Do (Not) Pause for Reconsideration of Critical Directives

The Noverify Mode (NOVERIFY) Directive causes the Noverify Mode to be entered. In Noverify Mode operator verification messages are not output. The file must be first initialized or substituted with Noverify Mode. (Refer to the Noverify Parameter of the INITIALIZE and SUBSTITUTE Directives.) If the attentive file is not initialized with Noverify Mode, this directive has no effect. This directive can be entered at any time, even before the FILE Directive is entered, and all subsequent FILE Directives enter the Noverify Mode if capable.

In Verify Mode, the following directives cause TOC/TOC32 to hold and display the following messages ('xxx' is a variable for a directive):

DIRECTIVE	MESSAGE
DELETE	DELETE OF xxx REQUESTED
INITIALIZE	INITIALIZATION OF xxx REQUESTED NOVERIFY WANTED
PECULIAR	PRIVILEGED STATE WANTED UNROLLABLE TASK WANTED
RENAME	RENAME OF MODULE WANTED
RESTORE	RESTORE REQUESTED
SUBSTITUTE	MINIMUM CATALOGER LEVEL BEING RAISED NOVERIFY WANTED

SYNTAX

NOV[ERIFY] [RES[ET]

[RES[ET]] - Parameter 1 (optional) causes the Verify Mode to be entered.

EXAMPLES

NOVERIFY	Enter the Noverify Mode if capable.
NOV RES	Leave the Noverify Mode; enter Verify Mode.

The Next Task or Overlay Cataloged Has System Option(s) Initialized When Loaded

The System Options (OPTIONS) Directive modifies the resource information in memory being collected for cataloging with the next task or overlay. The task's system options are examined when the task executes certain system services.

Special actions take place if a bit is set to ON (1). Each bit is treated as if it were a sense switch unique to the task. A System Operator Communication (OPT) Directive and REX services (GETOPT/MODOPT) permit other tasks to examine/modify the states of these switches, but normally only the operator task is used to change the task's system options. The CATALOG Directive permits the initial states of the individual switches to be set to any value (ON = 1 or OFF = 0) at the time the task or overlay is loaded.

Overlays are permitted to change individual bits of the task's system options, but they do not affect any task options if this statement is not specified for an overlay when it was cataloged. Unspecified bits are not changed. For root tasks, the initial state of all unspecified bits if off (0). Refer to Table 3-1 displaying the task's System Options.

SET (1) NAME		RESET (0) NAME	Bit in OPTION TCB Word	PURPOSE--if ON (1)
0		N00	0	}
1		N01	1	} Used by various processors
2		N02	2	} and FORTRAN run-time
3		N03	3	} package.
4		N04	4	}
5		N05	5	}
6		N06	6	Use 026 card conversion.
LO	or 7	NOLO or N07	7	Permit LO.
BO	or 8	NOBO or N08	8	Permit BO.
SC[RATCH]	or 9	NOSC or N09	9	Use SC file.
HO[LD]	or A	NOHO or NOA	10	Enter HOLD state at various breakpoints.
MA[P]	or B	NOMA or NOB	11	Permit map to be output.
GO	or C	NOGO or NOC	12	Continue even if errors exist.
AO	or D	NOAO or NOD	13	Use alternate file for commands.
HL[OAD]	or E	NOHL or NOE	14	Task holds after each load operation.
DU[MP]	or F	NODU or NOF	15	Dump body of task to global DO file if task aborts.

Table 3-1. Task's System Options

NOTE: In table 3-1, bits 7 to 15 have two names that can be used interchangeably. The numerical bit number forms (see column two) can be preceded with a character to allow entry in a form compatible with MAX III. (For example, U1 and NOU1 is the same as 1 and NO1.)

The rules of task compatibility, defined by the values of their respective influence limits, determine if one task is permitted to affect another task's system options.

SYNTAX

OPT[IONS] optioname [optioname...]
REMO[VE]

optioname
REMO[VE] - Parameter 1 (required) must be either optioname or REMOVE.

'optioname' can be set to the ON (1) or OFF (0) state.

Refer to Table 3-1. The default value of each bit is OFF (0) if its name is not specified for a root task. For overlays, all the ON (for example, LO) and OFF (for example, NOLO) states must be specified for option bits that are to be changed. Unspecified bits are not changed.

REMOVE specifies that the previous OPTIONS Directive is to be removed from TOC/TOC32's internal tables.

[optioname] - Parameters 2 through n (optional) allow the operator to set more than one system option in one directive statement.

EXAMPLES

OPTIONS DUMP,GO,SCRATCH,LO
Set DUMP, GO, SCRATCH, and LO options.

OPT REMO Delete the previous OPTIONS Directive from TOC/TOC32's internal tables.

OSPACE (TOC only)

Give the Next Two Map Task or Overlay Cataloged Extra Memory Space in the Operand Map

(TOC only.) The Operand Space (OSPACE) Directive allocates additional virtual memory in the task's operand space. OSPACE and its parameters are identical to ISPACE or SPACE Directives with the exception that all allocation references refer to operand space. When cataloging a one-map task or overlay, TOC uses the larger of either the ISPACE or OSPACE for the module.

SYNTAX

OSP[ACE] [allocated] [virtual]
[ALL] [ALL]

[allocated]
[ALL] - Parameter 1 (optional) can be either allocated or ALL.

'allocated' specifies the number of words of additional operand space to be immediately added. The default value is 0.

ALL specifies to allocate maximum (64K) operand space immediately.

[virtual]
[ALL] - Parameter 2 (optional) can be either virtual or ALL.

'virtual' specifies the amount of operand space to be allocated later at run-time.

ALL permits allocation of maximum operand space (64K) at run time.

EXAMPLES

OSPACE 512,#200	Add 512 words of operand space, 512 more words to be allocated at run-time.
OSP ALL	Allocate 64K of memory in operand space at load time.

The Next Program Cataloged is to be an Overlay

The OVERLAY Directive sets the Overlay Mode for subsequent cataloging operations. OVERLAY only recognizes the following directives that redefine optional traits of the next overlay cataloged:

PECULIAR REASSIGN VARIABLE OPTIONS POPTIONS	These directives optionally reinitialize selected data structures of the TCB of the host task when the overlay is loaded.	INSERT ERASE DEALLOCATE BIAS SPACE OSPACE	These directives reinitialize the preexisting virtual addressing space of the host task when the overlay is loaded.

These traits are collected in TOC/TOC32's resource table and are not active until the CATALOG Directive is executed. At catalog time, the information is converted into LPRs and cataloged with the body (MORs) of the overlay.

SYNTAX

OVE[RLAY] [name]

[name] - Parameter 1 (optional) is a 1- to 6-character alias name (CAN-code) by which the program is called for execution by the loader services. The name must be unique to all programs in the file being cataloged. If no name is specified, all six bytes of the actual program name, specified by the PROGRAM statement, contained within the object module chosen to be cataloged is used as the alias.

If MAIN is specified, the first three characters (bytes) of the program name is used as the alias name.

EXAMPLES

OVERLAY ANYPGM	Catalog as ANYPGM.
OVE UPDATE	Catalog as UPDATE.
OVERLAY	Catalog with all six bytes of program name.
OVE MAIN	Catalog with first three bytes of program name.

PAGESHARE

Allow The Task to Insert Global and Private Regions

The PAGESHARE Directive inserts page-sharing regions in the TCB of the next cataloged task (when that program is activated under MAX IV/MAX 32 as a task). The system uses these regions to manage the task's usage of global and private shared regions.

SYNTAX

PAG[ESHARE] maxnumber

maxnumber - Parameter 1 (required) specifies the maximum number of separate pageshare regions that the next cataloged task is allowed to insert at one time. The default value is 0. The value must be in the range of zero through 31.

EXAMPLES

PAGESHARE 2 Allow up to two shared regions.

PAG 10 Allow up to ten shared regions.

Display a Message and Wait for Response

The PAUSE Directive (same as ACTION Directive) outputs a message on the CO file directed to the user's attention. TOC/TOC32 is placed into the hold state, and a '/R' must be entered to continue. To abort TOC/TOC32 enter '/A'.

SYNTAX

PAU[SE] [text]

[text] - Parameter 1 (optional) displays a message directed to the user's attention. Default is the message 'PAUSE'.

EXAMPLES

PAUSE SET THE WRITE-PROTECT SWITCHES FOR DISC

The message 'PAUSE SET THE WRITE-PROTECT SWITCHES FOR DISC' is output to CO file.

PAU SET THE WRITE-PROTECT SWITCHES FOR DISC

The message 'PAU SET THE WRITE-PROTECT SWITCHES FOR DISC' is output to CO file.

PECULIAR

The Next Program Cataloged is Unusual

The PECULIAR Directive modifies the resource table in memory for the next task or overlay to be cataloged. It is only necessary if the task or overlay being cataloged requires special characteristics. (For example, to execute the next task or overlay in the privileged state or to catalog the next task as a special symbiont task). These special characteristics are entered as parameters in the PECULIAR Directive string. Each parameter is a keyword and one or more keywords can be specified in the same PECULIAR Directive string. To delete a previously specified characteristic, include the REMOVE parameter immediately following the parameter to be deleted (refer to examples).

Any keyword can be used for a task, but not all keywords can be used for an overlay (refer to Syntax section below).

CAUTION

To prevent any effect on the operating system, impose an appropriate system interlock on the batch task that is executing TOC/TOC32, so TOC/TOC32 can only catalog unprivileged and unrollable programs.

SYNTAX

PEC[ULIAR] AUT[OSTART],[REMO[VE]]
BAT[CH],[REMO[VE]]
CLM[EMORY],[REMO[VE]]
PIP[ELINE],[REMO[VE]]
NOP[IPELINE],[REMOVE]
PRI[VILEGED],[REMO[VE]]
REL[OCATABLE],[REMO[VE]]
REM[OTE],[REMO[VE]]
RIO[COMPLETE],[REMO[VE]]
SYM[BIONT],[REMO[VE]]
UNA[BORTABLE],[REMO[VE]]
UNR[OLLABLE],[REMO[VE]]

AUT[OSTART] - Parameters 1 (required) and 2 through 24 (optional).
BAT[CH]
CLM[EMORY]
PIP[ELINE] . If cataloging a task, can be any one or more of the keywords.
PRI[VILEGED]
REL[OCATABLE]
REM[OTE] . If cataloging an overlay, can be one or more of the following: CLMEMORY, PIPELINE, PRIVILEGED, RELOCATABLE, and/or RIOCOMPLETE.
RIO[COMPLETE]
SYM[BIONT]
UNA[BORTABLE]
UNR[OLLABLE]

The REMOVE keyword indicates the opposite effect (for example, 'PECULIAR PRIVILEGED REMOVE' causes the next program to be unprivileged). REMOVE is useful when a SOURCE,,INIT statement is specified and some of the PECULIAR parameters need to be changed.

AUTOSTART causes the next cataloged task to automatically activate when the system is first started, providing the BATCH causes the task to reload and transfer control to an overlay everytime the task exits or aborts. The overlay that reloads is the root task itself. The overlay can be cataloged for each host task as a task under a unique alias name or a task can be duplicated with multiple, unique resources and all can share a single overlay (refer to the DUP Directive). Memory requirements of a task are expanded or reduced to satisfy the needs of Job Control.

CLMEMORY causes the next cataloged task's or overlay's memory that has been allocated for its body (the memory between the program's bias and high limit) to be cleared to 0 before the program is loaded into memory.

PIPELINE causes the next cataloged program to enter the Pipeline Mode when it is executed. Pipeline Mode shortens run time. A program cannot execute in Pipeline Mode if it stores instructions or operands inline within eight instructions of the currently executing instruction.

NOPIPELINE (TOC32 only.) Pipeline mode is the default case for TOC32 overlays and tasks. This command enables users to defeat this mode of operation.

PRIVILEGED causes the next task or overlay to be cataloged so that it executes in the privileged state when activated or loaded. Only use this parameter for a fully debugged task that must execute privileged instructions. The risk is augmenting or destroying the operating system. (Even though a privileged task has isolated virtual addressing space, it can select MAP 0 as its operand space, and can execute final instructions (such as HLT) and interrupt control instructions.) A privileged overlay executes in the Unprivileged Mode unless its root task is privileged. This parameter cannot be used if TOC/TOC32 is operating in Unprivileged Mode.

RELOCATABLE causes the program to be relocated at load time if a run-time bias is specified when the loader is called.

The BIAS Directive, if used, adds additional bias to the run-time bias. This program cannot be used with programs with no relocatable object. If RELOCATABLE is not specified when the program is cataloged the relocation bits are lost, and RELOCATABLE cannot be added to a PUT,,NEW Statement. TOC32 relocatable programs are limited to a size of #80000 16-bit words.

REMOTE allows the next cataloged task to respond during a hold state to the RESUME and ABORT Directives from the remote terminal (as long as CO and OC are assigned to the remote terminal).

RIOCOMPLETE causes the next cataloged task or overlay to enter the Resume I/O Complete Mode. This sets the R:IO flag in the TCB of the task loading this module. The current R:IO state of a task is defined by the RIOCOMPLETE resource of the last module loaded.

SYMBIONT allows the next cataloged task to simulate imaginary I/O devices (such as a spooler task). This links the task to its controller table that contains the I/O queues provided and maintained by the Basic I/O System (BIOS) for the task's imaginary devices. A symbiont can be nonresident and loaded only when the BIOS has operations queued to its controller. A task can specify criteria for becoming nonresident since all I/O operations queued for it are in a resident table not associated with the body of the task.

UNABORTABLE causes the next cataloged task to be automatically reactivated whenever it exits or aborts.

UNROLLABLE (TOC only) allows the next cataloged task to not release any of its memory resources until it exits, even to higher priority tasks. This feature should not be used except for tasks that must remain in memory at all times (to acquire data or function in a timely manner). Overlays cannot invoke this parameter, but they are unrollable if their root task is unrollable.

EXAMPLES

PECULIAR PRIVILEGED
Catalog the next task or overlay so that it executes as privileged when activated or loaded.

PEC PRIVILEGED,REMOVE
Catalog the next task or overlay so that it executes as unprivileged when activated or loaded.

PEC SYMBIONT Catalog the next task as a special symbiont task.

PEC AUT,REMO,REL,REM
Remove AUTOSTART and add RELOCATABLE and REMOTE.

The Next Task or Overlay Cataloged has Program Options Initialized When Loaded

The Program Options (POPTIONS) Directive modifies the resource information in memory being collected for the next task or overlay to be cataloged. The task's program options are kept in a 16-bit word in its TCB that the system never examines. Each bit is treated as if it were a sense switch unique to the task.

Combining the Program Options Directive (POP) and REX services (GETPOP/MODPOP) permits other tasks to examine or modify the states of these switches in any task. The POPTIONS Directive permits the initial states of these individual switches to be set to any value (ON = 1 or OFF = 0) at the time the task or overlay is loaded. The name of each switch is based on the bit number assigned to it in the program option word (refer to Table 3-2 below). Overlays are permitted task options if this statement is not specified for an overlay.

Unspecified bits are not changed. For root tasks, the initial state of all unspecified bits is OFF (0). Refer to Table 3-2 displaying Task's Program Options.

ON (1) NAME	OFF (0) NAME	Bit in TCB OPTION Word
0	NO0	0
1	NO1	1
2	NO2	2
.	.	.
.	.	.
.	.	.
9	NO9	9
A	NOA	10
B	NOB	11
C	NOC	12
D	NOD	13
E	NOE	14
F	NOF	15

Table 3-2. Task's Program Options

The rules of task compatibility, defined by the values of their respective influence limits, determine if one task is permitted to affect another task's program options.

SYNTAX

POP[TIONS] optioname [optioname...]
NO[optioname]
REMO[VE]

optioname
NO[optioname]
REMO[VE] - Parameter 1 (required) must be either: optioname, NOoptioname, or REMOVE.

'optioname' is the name of any program bit to be set to ON (1). The default for a root task is OFF (0); the default for overlays is unchanged.

'NOoptioname' is the name of any program bit to be set to OFF (0).

REMOVE specifies that the previous POPTIONS Directive is to be removed from TOC/TOC32's internal tables.

[optioname] - Parameters 2 through n (optional) allow the operator to enter more than one option in the directive string.

EXAMPLES

POPTIONS 6 Sets Bit 6 to ON.

POP 0,1,B,NO2,NOF
Sets bits 0, 1, and 11 to ON; resets bits 2 and 15 to OFF.

POP REMO Delete the previous POPOPTIONS Directive from TOC/TOC32's internal tables.

Recatalog a Program Retrieved by GET Directive

The PUT Directive causes a program that was previously retrieved by the GET Directive and placed on a sequential save file to be recataloged either onto the original directoried load-module file (where GET retrieved it) or onto another load-module file. The PUT Directive gives the operator the choice of either recataloging the program as is or changing the resources while recataloging.

(For example, making an overlay into a task or adding new files to a task.)

PUT only accepts object generated by the GET Directive. PUT cannot operate on a privileged module if TOC/TOC32 is unprivileged. TOC/TOC32 uses the largest space available to catalog this module. If PUT is preceded by the TMTCB Directive, then a two-map TCB is created for the recataloged task. Refer to Chapter 4 that displays a recataloging job stream and refer to Figure 3-7 and Figure 3-8.

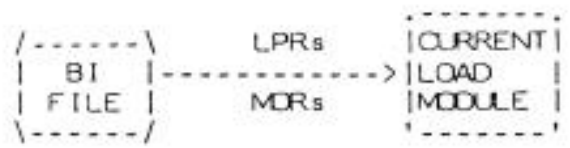


Figure 3-7. Schematic of PUT Directive with Identical Resources

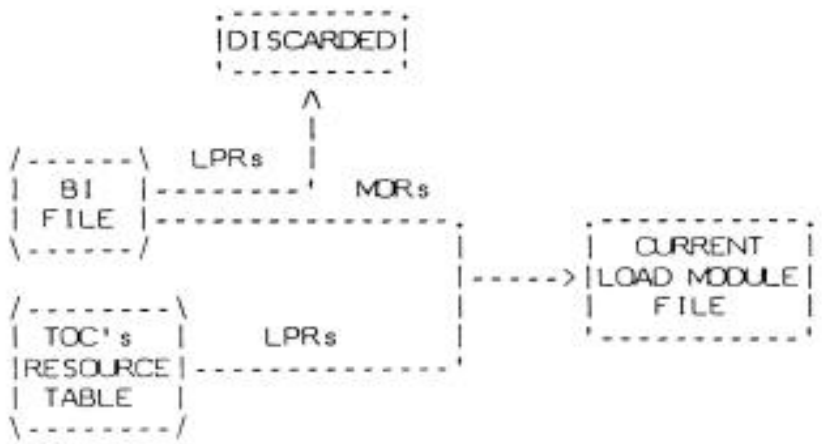


Figure 3-8. Schematic of PUT Directive with New Resources

SYNTAX

PUT [pgmname] [NEW[RESOURCES]]

[pgmname] - Parameter 1 (optional) specifies the 1- to 6-character name of the program to be recataloged. This is the name by which it was known when it was retrieved by the GET Directive. If not specified, the first program encountered on the sequential BI file is recataloged.

[NEW[RESOURCES]] - Parameter 2 (optional) causes the resource records (LPRs) of the original program to be discarded. Then new LPRs are synthesized from the resource tables in TOC/TOC32 and cataloged with the program. (Refer to Figure 3-8.) Note that the recataloged program is named according to the alias name specified on the TASK or OVERLAY Directive when NEWRESOURCES are specified. If this parameter is not specified, the recataloged program is identical to the original including the name; however, once it has been recataloged, the RENAME Directive can be used to rename it.

EXAMPLES

PUT	Recatalog first program under old name with same resources.
PUT MAIN	Search for MAIN and recatalog it with same resources.
PUT,,NEW	Recatalog first program with new resources.
PUT B,NEW	Search for B and recatalog it with new resources.
TMTCB PUT	Recatalog first program under old name with the same resources. If it is a task, create a two-map TCB.

REASSIGN

Reassign the Logical File Name if the Load Module is Loaded as an Overlay

The REASSIGN Directive defines or redefines, repeatedly if necessary, logical files of the host task when an overlay is loaded by the host task. REASSIGN is only appropriate when cataloging overlays or tasks that might be reloaded as an overlay. A module cataloged with a REASSIGN Directive changes the specified logical file assignments when loaded. Any logical files not specified by a REASSIGN Directive are not affected.

SYNTAX

REA[SSIGN] logfilename normal [default][TEM[PORARY]]
REMO[VE]

logfilename - Parameter 1 (required) specifies a 1- to 3-character CAN-codable name of a logical file to be either reassigned or defined if a vacant logical file entry exists in the host task.

normal
REMO[VE] - Parameter 2 (required) must be either normal or REMOVE.

'normal' is a 1- to 3-character CAN-code name of a system device or real file to be assigned to the logical file. It can be the name of another logical file that is already defined.

REMOVE specifies that REASSIGN is to be removed from TOC/TOC32's internal tables. This parameter is useful when a REASSIGN is to be changed or removed from the resources of a task (Refer to the SOURCE Directive) or if a typing error is made when entering a REASSIGN. If an incorrect entry is not removed it remains in TOC/TOC32's internal tables.

[default] - Parameter 3 (optional) is the 1- to 3-character CAN-code name of a system device or logical file to be assigned to a logical file. The default assignment is an unused file assignment. Batch tasks use this file to reset the normal assignment after a $JOB statement is processed. Overlays loaded under Job Control are not affected by this parameter.

[TEM[PORARY]] - Parameter 4 (optional) specifies that the REASSIGN is temporary and can be removed by the CTASSIGNMENTS Directive when another module is loaded.

EXAMPLES

REASSIGN LO,LPP	Reassign LO to line printer.
REA 27,MT1,MT2	Reassign logical file 27 to magnetic tape unit 1; default to magnetic tape unit 2.
REA 27,REMO	Remove prior assignment of logical file 27.
REA 27,MT1,MT1,TEM	Temporarily reassign logical file 27 to magnetic tape unit 1; default to magnetic tape unit 1.

REMOVE

Delete a Cataloged Task or Overlay From the Current File

The REMOVE Directive eliminates a specified task or overlay program from the directory of the selected load-module file. REMOVE can be executed at any time.

If the name to be removed is a duplicate, this is the only action taken. If the name is for an original cataloged program, TOC/TOC32 deletes the space used by the program and all duplicates are recovered during the next COMPRESS operation to the file. Duplicate masters (oldname) must be recataloged before a COMPRESS or SAVE can be done. The DELETE Directive performs the same function as REMOVE.

The space used by this module is reused when a new program is cataloged and this space is a part of the largest space available.

SYNTAX

REM[OVE] name

name - Parameter 1 (required) specifies the alias name that was cataloged with the task.

EXAMPLES

REMOVE FRANK Remove program FRANK from the directory.

REM BRENT Remove program BRENT from the directory.

Change the Name of a Cataloged Task or Overlay

The RENAME Directive changes the name of a cataloged load module in the directory. No other changes are made to the task or overlay. RENAME can be entered at any time.

NOTE: This Directive was previously called NAME. NAME is still supported for existing procedures, but from now on use the RENAME Directive instead because support for the NAME Directive can be deleted in the future.

An example of the use of this directive is for using existing Job Control procedures. For instance, if a program is listed as M5A in the directory and the procedure calls the program MAC, use the RENAME Directive to change the oldname 'M5A' to 'MAC' in the directory.

SYNTAX

REN[AME] oldname newname

oldname - Parameter 1 (required) specifies the name now in the directory that is to be replaced.

newname - Parameter 2 (required) specifies the name that replaces the old name.

EXAMPLES

RENAME MEH,MEC	Change the program name of MEH to MEC in the directory.
REN M5ASM,M5A	Change the program name of M5ASM to M5A in the directory.

RESTORE

Reconstruct the Current Load Module File from the Sequential Save-Format File (BI)

The RESTORE Directive reads from the BI file and causes the load module output to another file by a SAVE or COMPRESS Directive to be placed back either on the current file or a new file. RESTORE inputs only the output of the SAVE or COMPRESS Directives. RESTORE can restore (place back, reconstruct) a file with a sector length different from that of the saved file, but the restored file is not as efficient in disc space as the file is when cataloged in its entirety.

The BI file must be manually positioned to the desired save file. Records are read from the BI file and used to initialize and completely restore the selected load-module file, building a new directory, and writing LPRs and MORs into contiguous areas of the file medium. A file mark being read from BI terminates the process.

The RESTORE statement requires operator intervention if the load-module file is in Verify Mode. (Refer to INITIALIZE Directive). Using RESTORE is equivalent to using both INITIALIZE and CATALOG. RESTORE does not operate if TOC/TOC32 is operating in Unprivileged Mode (refer to the PECULIAR Directive, UNPRIVILEGED Parameter).

SYNTAX

RES[TORE]

EXAMPLES

RESTORE	Reconstruct the current load-module file from the sequential SAVE or COMPRESS format BI file.
RES	Reconstruct the current load-module file from the sequential SAVE or COMPRESS format BI file.

Rewind Device to Beginning-of-Medium

The REWIND Directive positions a device or devices to their beginning-of-medium (BOM) position(s). If the device(s) cannot rewind, either the directive is ignored or TOC/TOC32 performs a normalization function on the device(s). The File Position Index (FPI) in the File Assignment Table resets to 0.

SYNTAX

REW[IND] file [file...]

file - Parameter 1 (required) specifies the logical file to be rewound.

[file] - Parameters 2 through n (optional) allow the operator to specify more than one file in the directive string.

EXAMPLES

REW BI,BO Rewind logical files BI and BO.

REWIND BO Rewind logical file BO.

SAVE

Copy Contents of the Load-Module File onto a Sequential Save-Format File (BO)

The SAVE Directive copies an entire load-module file, with directory, onto the BO file (BO must be manually positioned by using REWIND, AVFILE, and BKFILE Directives) in a sequential save format. The directory, LPRs, and MORs of the entire file (less deleted programs and directory vacancies) are written on the sequential BO file. Records are broken into pieces suitable for the sequential BO file. Vacant directory entries and deleted load modules are not saved. An End-Of-File (EOF) mark is automatically written on the BO file at the end of each save.

CAUTION

The SAVE Directive does not operate if there are duplicate masters that have been removed, unless either:

- The duplicate master is recataloged

or

- The duplicate is also removed

The space taken by the saved file can be slightly larger than the space taken by the active load-module file because of extra control words written in the save format. SAVE only operates on a privileged module if the TOC/TOC32 is privileged.

Refer to COPY Directive for moving the SAVE output to other devices.

SYNTAX

SAV[E]

EXAMPLES

SAVE	Copy entire contents of current load-module file onto a sequential-save file (BO).
SAV	Copy entire contents of current load-module file onto a sequential-save file (BO).

List the Source of the Resources for this Program

The SOURCE Directive generates a list of the resources to LO and optionally copies the source of the resources to a specified file, or initializes TOC/TOC32's internal tables with the resources of the specified task or overlay. Not all resources appear exactly the way they were first entered (values are output in hexadecimal notation; some values are rounded to page boundaries). This directive does not process INSERTs with security keys since the integrity of the system can be destroyed.

An EOF mark is written to the specified file after the resources of Parameter 1 have been output. (Refer to Figure 3-9.)

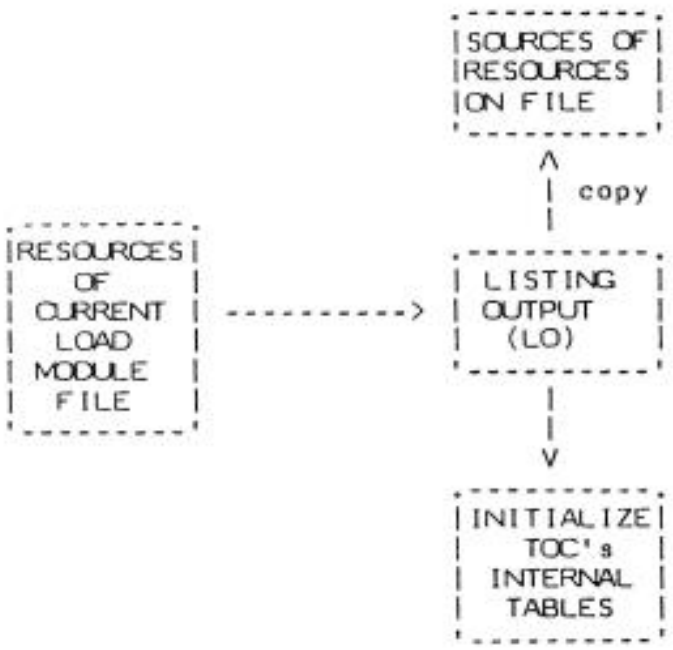


Figure 3-9. Schematic of SOURCE Directive

SYNTAX

SOU[RCE] name [file]
[INIT[IALIZE]]

name - Parameter 1 (required) specifies the name of a program already cataloged on the currently specified load-module file.

[file] - [INIT[IALIZE]] Parameter 2 (optional) can be either file or INITIALIZE.

'file' is the name of a logical file where resources are output. If no file is given, TOC/TOC32 only lists the resources of 'name' to the LO file.

INITIALIZE specifies that TOC/TOC32's internal tables are to be initialized. TOC/TOC32 initializes these tables with the resources of 'name' as previously cataloged.

EXAMPLES

SOURCE B,BO	Give a report on LO and the actual resources on BO of the Batch Task B.
SOURCE S,SO	Give a report on LO and the actual resources on SO.
SOU TOC	Give only a report of the resources of TOC/TOC32 on LO.
SOU OVTOC	Give only a report of the resources of TOC/TOC32's overlay OVTOC on LO.
SOU B INIT	Initialize TOC/TOC32's tables with the resources of B and give a report on LO.

Give the Next Task or Overlay Cataloged Extra Memory Space

The SPACE Directive (same as ISPACE Directive) makes the next cataloged task allocate more virtual instruction space than is evident from its natural size and relocation bias. Optionally, it allocates additional private memory space to all or part of this additional instruction space before the program is loaded. If SPACE or ISPACE is not entered, TOC/TOC32 loads the next task or overlay into memory space assumed to already exist or space just sufficient to hold the program body after any optional bias. SPACE permits additional allocated or unallocated memory space to be bound to the program, above the natural program, when it is loaded. The allocation takes into account a bias (refer to Figure 3-10). Only a task can request additional virtual-memory space. An overlay can only allocate additional space within the virtual space limits already owned by its host task. A series of REX services can be used by a task or overlay to assign actual memory to unallocated space at execution time.

The Allocated Parameter is useful for data chaining when the transfer counts are too small on fast discs. (Refer to the description on the GAP Directive before reading further.) If GAP cannot be used or if backward or absolute transfer addresses are encountered, use the allocated parameter to increase load-time efficiency with a minimum of space being allocated.

If a one-word Transfer Count (TC) is encountered while cataloging on a fast disc, a dummy read is placed after the one word TC so that another TC of one word can follow. The Allocated Parameter must be specified so that TOC/TOC32 can give the Load-Module Loader a buffer to read into. SPACE allocates an entire page (256 words) even if the Allocated Parameter specifies just one word.

Refer to the BLOCK Directive for additional information about overriding default values for data chaining.

NOTE: Use the appropriate syntax below for TOC or TOC32.

SYNTAX (for TOC)

```
SPA[CE]        [allocated]    [virtual]
               [ALL]          [ALL]
```

[allocated] - [ALL]
Parameter 1 (optional) can be either allocated or ALL.

'allocated' is the number of words of allocated memory to be given to the program in addition to the memory size allocated for its natural instruction space. The default value is 0.

ALL (valid for TOC only) specifies that the rest of virtual instruction space is wanted. This parameter is useful for programs like Sort/Merge that need all available memory to run efficiently.

[virtual]
[ALL] - Parameter 2 (optional) can be either virtual or ALL.

'virtual' is a total of the number of words of unallocated virtual addressing space plus all natural and allocated instruction space. The default value is 0. Only tasks can use this parameter.

ALL (valid for TOC only) specifies that the next cataloged task is able to have a full instruction map (up to 64K of memory).

SYNTAX (for TOC32)

SPA[CE] [allocated pages] [virtual pages]
[AL[L]1]
[AL[L]2]
[AL[L]3]

[allocated pages] - Parameter 1 (optional) represents the number of additional memory pages to be allocated at load time following the programs natural memory allocation.

[virtual pages]
[AL[L]1]
[AL[L]2]
[AL[L]3] - Parameter 2 (optional) represents the number of available pages that can be dynamically allocated through program control.

AL1, AL2, AL3 (valid for TOC32 only) specifies MAP image space for a first level map (AL1), a second level map (AL2), or a third level map (AL3).

EXAMPLES (for TOC only)

SPACE 500,500	Add 500 words of instruction space, 500 more words that can be allocated at run-time.
SPACE #1000	Add 4096 words of allocated space.
SPACE 0,#400	Allow up to 1024 more words to be allocated at run-time (by the task).
SPACE ALL	Allocate maximum memory.
SPACE 0,ALL	Permit allocation of up to 64K of memory at run time.

EXAMPLE (for TOC32 only)

SPA 3,10	Allocate an additional 3 pages of memory at load time, and leave map image room for an additional 10 pages of memory to be allocated dynamically under program control.

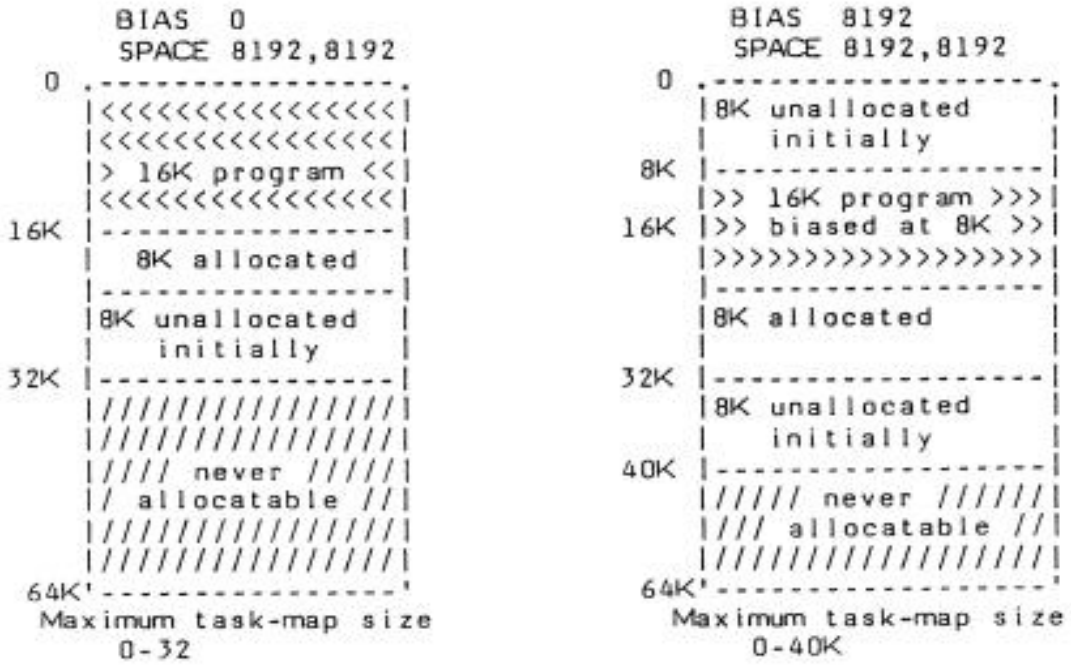


Figure 3-10. Effects of SPACE Directive on Memory Bias

SQUEEZE

Consolidate as Much Free Space as Possible

The SQUEEZE Directive can be entered at any time. It moves load modules one at a time to consolidate as much free space together as possible. This directive is an automated sequence of the GET, DELETE, and PUT Directives. Directory sectors in the middle of the load module file are not moved. Therefore, holes in the load module file could exist after a SQUEEZE. These are eventually used when modules are added or replaced. Enough directory sectors can be created at the beginning of the file using the INITIALIZE Directive option.

If a system failure occurs during a SQUEEZE request, the load module is still in the file or on the scratch file specified in the command. See also COMPRESS Directive.

SYNTAX

SQU[EEZE] file

file - Parameter 1 (required) specifies the logical file name of a scratch file and must be large enough to hold individual load-module entries being compressed in the save format.

EXAMPLES

SQUEEZE SC Compress load-module file using scratch file SC.

SQU WRK Compress load-module file using scratch file WRK.

NOTE: This directive is not supported on discs that cannot handle cylinder boundary overflow.

The Task Uses an Unusual Sized Stack

The STACKS Directive (TOC only) defines the initial size of two Last-In-First-Out (LIFO) stacks in the TCB of the next cataloged task at the time it is activated as a task under MAX IV. These stacks determine the task's ability to nest executive services and are extended dynamically as usage dictates.

The main stack is extended and reduced by 256 additional words (beyond its initial size) as usage dictates. The other stack need not be as large and is fixed in size by STACKS.

SYNTAX

STA[CKS] [bigwords] [littlewords] [EX[TENSIBLE]] [NE[XTENSIBLE]]

[bigwords] - Parameter 1 (optional) is the initial size of the next cataloged task's main register-save LIFO stack. The default value is 100 words, although TOC allocates and deallocates up to 256 additional words automatically if usage dictates.

If set to 0, the system allocates a fixed-size stack of 256 words during execution of a task that is not dynamically extended or restricted. The value of Parameter 1 must be in the range 0-255.

[littlewords] - Parameter 2 (optional) is the fixed size of the next cataloged task's special PS/PR stack used in nesting of REX services. The default value is 8 words that permit up to three nesting levels of REX services calling other REX services.

[EX[TENSIBLE]] [NE[XTENSIBLE]] - Parameter 3 (optional) can be either EXTENSIBLE or NEXTENSIBLE.

EXTENSIBLE allows the main stack to be automatically extended up to an additional 256 words if the main stack's usage exceeds the initial size. Default is EXTENSIBLE.

NEXTENSIBLE inhibits the main stack from being automatically extended.

EXAMPLES

STACKS 0,6	The big stack has a maximum of 256 words (extensible). The small stack has 6 words.
STA 75,10	The big stack has a maximum of 331 words (extensible). The small stack has 10 words maximum.
STA 255,12	The big stack has a maximum of 511 words (extensible). The small stack has 12 words maximum.
STA 120,8,NE	The main stack has a maximum of 120 words. The little stack has 8 words maximum.

Change the File's Security Key and/or Limit

The SUBSTITUTE Directive changes the security lock, file influence limit, and other file-control options of the currently selected load-module file as specified in the INITIALIZE Directive. Existing security characteristics must be passed before being changed. For example, the previous security key (if any) must be entered by the KEY Directive before a new key can be specified. In this case, the new key, once accepted, does not have to be entered on a subsequent KEY Directive as long as this file is used for the current load of TOC/TOC32.

To remove a key from a file refer to KEY Directive or Parameter 1 below.

SYNTAX

SUB[STITUTE]	KEY	[newkey]
	LIM[IT]	newlimit
	NOV[ERIFY]	
	VER[IFY]	

KEY
LIM[IT]
NOVER[IFY]
VER[IFY] - Parameter 1 (required) must be either KEY, LIMIT, NOVERIFY, or VERIFY.

KEY defines a new security key that must be used to open a file. In Parameter 2, 'newkey' specifies the new key name.

LIMIT along with 'newlimit' (refer to Parameter 2) specifies a new influence limit for the file.

NOVERIFY inhibits the operator from verifying cataloging functions.

VERIFY allows the operator to verfiy cataloging functions.

[newkey]
newlimit - If KEY is specified for Parameter 1, Parameter 2 (optional) can be newkey. If LIMIT is specified for Parameter 1, Parameter 2 (required) must be newlimit. If VERIFY or NOVERIFY is specified for Parameter 1, Parameter 2 is not used.

'newkey' specifies the new security key name to open a file (providing KEY is specified for Parameter 1). (Refer to KEY Directive for description of legal keys.)

'newlimit' specifies the new minimum cataloger influence limit for a file. If 255 is entered, any cataloger can open a file.

EXAMPLES

SUBSTITUTE KEY PQR	Change to new security key PQR.
SUB KEY	Remove the security key.
SUB LIMIT 240	Minimum cataloger influence limit that can access this file is now 240.
SUB LIM 255	Any cataloger influence limit can now access this file.
SUB NOV	Inhibit the operator from verifying cataloging functions.
SUB VER	Allow operator to verify cataloging functions.

Specifies Maximum Number of System Pages to be Allocated

The System Pages (SYSPAGES) Directive specifies the maximum number of pages that can be allocated in the executive by system services. TOC/TOC32 aborts the task if system services attempt to allocate more system pages than are specified by Parameter 1. The maximum number of pages does not affect the allocation of private pages. Some of the system services affected by this maximum number of system pages are:

- Any service that requires the allocation of a service buffer.
- A push that requires the extension of a stack.
- The pages required by the file manager for the generation of FCBs.
- Temporary buffers for the file manager for the passing of parameters.

If the SYSPAGES Directive is not specified, the default number of system pages is set to 2. This default value should be adequate for all system services except the MAX IV or MAX 32 File Manager services.

SYNTAX

SYS[PAGES] numpages

numpages - Parameter 1 (required) specifies the maximum number of pages that can be allocated.

EXAMPLES

SYSPAGES 10	Set the maximum number of pages that can be allocated by the total system to 10.
SYS	Set the maximum number of pages that can be allocated by the total system to default (2).

TASK

The Next Program Cataloged is a Task

The TASK Directive declares that the next program to be cataloged is a task. TASK sets Task Mode and enables other directives that redefine optional characteristics.

SYNTAX

TAS[K] [name] [priority] [limit]

[name] - Parameter 1 (optional) is a 1- to 6-character CAN-code alias name by which the task is called for execution by the ACTIVATE, ESTABLISH, CONNECT, or other task-scheduling services. Refer to the discussion of REX Services in the MAX IV or MAX 32 GENERAL OPERATING SYSTEM System Guide Manual. The name must be different than the other programs in the file.

If the name is MAIN, the first three characters of the program name are used as the alias. Default is the actual program name defined within the object module to be cataloged.

[priority] - Parameter 2 (optional) is the execution level (0 to 255) at which the task should initially enter the CPU queue when activated or scheduled. The highest execution priority is 0 and the lowest execution priority is 255. A default is specified by the task that schedules or activates the task.

[limit] - Parameter 3 (optional) is the task influence limit (0 to 255) this task has over other tasks. No task with a lower influence limit can perform a service upon this task and this task cannot affect tasks with a higher influence limit.

The highest influence level is 0. The lowest is 255. The task's execution priority is never allowed to exceed the influence limit. Default is priority level (Parameter 2).

EXAMPLES

TASK	Catalog using actual program name.
TASK MAIN	Catalog using first three characters of actual program name.
TAS,,35	Catalog using actual program name with priority 35.
TAS ABCPQR	Catalog using alias name ABCPQR.
TAS A,72,28	Catalog Task A with priority 72 and influence limit 28.

TEST

Check the Integrity of BI File

The TEST Directive reads and verifies the contents of a sequential standard binary file (such as a load-module file generated by the SAVE Directive) to ensure integrity (format, checksums, and sequence). Manually position the BI file prior to executing the TEST Directive. (Refer to REWIND, AVFILE, and BKFILE Directives.)

The TEST Directive is used to test the integrity of any standard binary-object file (for example, output from an assembler or link editor). TEST reads data until a file mark is detected.

SYNTAX

TES[T]

EXAMPLES

TEST	Check the integrity of all records in the sequential save-format file BI.
TES	Check the integrity of all records in the sequential save-format file BI.

Limit the Time the Next Task Cataloged has to Execute

The TIMELIMIT Directive defines the maximum CPU time the next cataloged program can consume during execution by modifying the resource information. TOC/TOC32 gives unlimited execution time to a task if no time limit is specified or if the TIMELIMIT statement indicates 0 minutes, and 0 ticks. A single numeric parameter indicates a tick specification.

Once an overlay of a task sets a limit, the limit applies to the host task. A subsequent overlay must select a new limit or select no limit (0,0) to ensure continuation of the task. The time limit specified is for actual CPU time consumed by the task's programs and not elapsed clock time. The accuracy of this limit is plus or minus one tick. TOC/TOC32 aborts a task that exceeds a set limit, or prints a message indicating the timeout.

SYNTAX

```
TIM[ELIMIT]    minutes    ticks     [ABORT]
               REMO[VE]   [ABORT]
```

minutes
REMO[VE] - Parameter 1 (required) must be either minutes or REMOVE.

'minutes' is the maximum number of minutes of actual CPU execution time this task is be permitted to execute. The default value is 0.

REMOVE specifies the TIMELIMIT previously entered is to be removed from TOC/TOC32's internal tables. This parameter is useful when the TIMELIMIT is to be removed or changed in the resources of a task. Incorrect entries must be removed.

ticks
[ABORT] - Parameter 2 (optional) can be either ticks or ABORT.

'ticks' is the fractional part of a minute that the task is permitted to execute. The resolution of this value depends on system generation and is a multiple of the clock (.005 second). The default value is 0.

ABORT ends the task if the time limit expires.

[ABORT] - Parameter 3 (optional) ends the task if the time limit expires.

EXAMPLES

TIMELIMIT 3,3000	Time limit is 3 minutes and 30 seconds (.01 second per tick).
TIM 2,ABORT	Time limit is 2 minutes; if time limit expires, abort the task.
TIM REMO	Remove the time limit.

This Task Is To Have A Two-Map TCB

The Two-Map Task Control Block (TMTCB) Directive (TOC only) defines a task as having a two-map TCB although the task might have been link-edited to execute with one-map.

Use TMTCB, for example, to catalog or recatalog a one-map task, such as the background Job Control task, to have it load and execute both one- and two-map overlays.

SYNTAX

TMT[CB] [REMO[VE]]

REMO[VE] - Parameter 1 (optional) removes the two-map specification.

EXAMPLES

TMTCB	Specify a two-map TCB.
TMTCB REMOVE	Remove two-map specification.

UNIX (TOC32)

This Task Is To Have A UNIX Extension

The UNIX Directive (TOC32 only) designates that the next task catalogued is to have a UNIX extension included (or removed) in its TCB resources.

SYNTAX

UNI[X] [REMO[VE]]

REMO[VE] - Parameter 1 (optional) removes the UNIX resource.

EXAMPLES

UNI	This task is to have a UNIX extension.
UNI REMO	Remove the UNIX extension.

Add a User Extension to the Next Task

The USER Directive adds a user extension to the EXT of the next task cataloged. Refer to the MAX IV or MAX 32 DATA STRUCTURES, System Design Manual for information about the user extension (EXTUSR).

SYNTAX

USE[R] number

number - Parameter 1 (required) indicates the number of words to add. The range is 0 through 255. Selecting 0 causes removal of the user extension.

EXAMPLES

USER 12	Add a user area of 12 words.
USE 0	Remove the user area.

VACATE

Make a Vacant Entry in the Directory

The VACATE Directive inserts a vacant entry at a specific point in the directory of a specific load-module file. When executed, the specified entry number in the directory is made vacant and used for the first vacant entry if the name being cataloged is not already in the directory. All other entries in the directory are pushed down to make room for the vacant entry. The LIST Directive specifies the entries in the directory in exactly the same order as they are positioned in the directory.

VACATE locates entries in a file's directory so that the most important programs can be maintained near the beginning of the directory regardless of the order in which they were cataloged. This is particularly important when the file is to be used with a resident directory that is smaller than the file's disc-resident directory. In this case, the directive allows the user to move the most important programs into the range of the resident part (first part) of the directory.

SYNTAX

VAC[ATE] entrynumber

entrynumber - Parameter 1 (required) is the numerical position in decimal of the entry in the directory (1 is the first).

EXAMPLES

VACATE 27 Make entry number 27 vacant.

VAC 1 Make the first entry number 1 vacant.

Define or Change the Value of this Variable

The VARIABLE Directive is used to set up and define values for variables of the next task or overlay to be cataloged. VARIABLE can be repeated any number of times.

A task can define an unlimited number of specified and unassigned variables but an overlay can only add variables where vacancies exist or change the values of existing variables.

An Operator Communication (OC) Directive called LET has the capability to examine any task's variables. Also, several REX services such as GETVAR or SETVAR allow tasks to communicate with these specified variables.

The unspecified variable is the variable that TOC/TOC32 uses if TOC/TOC32 cannot find the named variable. The unspecified variable cannot be named by an overlay.

SYNTAX

```
VAR[IABLE]      [name]      [value]
                [number]    REMO[VE]
                [REMO[VE]]
```

[name]
[number]
[REMO[VE]] - Parameter 1 (optional) can be either name, number, or REMOVE. If it is absent, the VARIABLE Directive selects an unspecified

'name' is the 1- to 3-character CAN-code variable name to be revalued or defined in the task's TCB. If name is selected for Parameter 1, value must be specified for Parameter 2 (refer to the following decription of value).

'number' reserves vacant space for number of variables.

REMOVE, if specified as Parameter 1, deletes all variables from TOC/TOC32's internal tables.

[value]
[REMO[VE]] - If name is selected for Parameter 1, Parameter 2 is required and must be 'value'. If number is selected for Parameter 1, Parameter 2 is optional and can be either 'value' or REMOVE.

'value' is the decimal, hexadecimal, or CAN-code value to be assigned to the new existing variable.

REMOVE specifies that the variable of Parameter 1 is to be deleted from TOC/TOC32's internal tables. REMOVE is useful when a variable is to be deleted, given a different value, or if a typing error is made. If the variable is not removed it remains in TOC/TOC32's internal tables.

EXAMPLES

VARIABLE DELAY,50	Set variable DELAY to 50.
VAR PHASE,-6	Set variable PHASE to -6.
VAR MASK,#F0F0	Set variable MASK to #F0F0.
VAR DELAY,REMO	Remove variable DELAY.
VAR	Select unspecified variable.
VAR 7	Reserve vacant space for 7 variables.

Write an End-Of-File Mark

The Write End-Of-File (WEOF) Directive writes an End-of-File (EOF) mark on each selected logical file.

For some devices, this is a hardware function; for others, a software equivalent is devised and detected by handlers during reading or positioning operations for standard data formats. Further information about device-handler operations can be found in the appropriate MAX IV or MAX 32 handler manual listed in the Preface.

CAUTION

Entering the WEOF Directive with the name that specifies the load-module file directory causes data loss.

SYNTAX

WEO[F] file [file...]

file - Parameter 1 (required) specifies the name of the logical file on which the EOF is written. If not specified, the default is no action.

[file] - Parameters 2 through n (optional) are other files to which EOF marks are to be written.

EXAMPLES

WEOF BO Write one EOF on logical file BO.

WEO BO,BI Write EOFs on logical files BO and BI.

WORKBENCH

Add Room for the Workbench Extension to the Next Task Cataloged or PUT

CAUTION

This directive is only compatible with Revision G.1 (or later) of the MAX IV Operating System or Revision A.0 (or later) of the MAX 32 Operating System. To prevent data loss, do not use the WORKBENCH Directive if using Revision G.0 of the MAX IV Operating System.

The WORKBENCH Directive causes the workbench extension to be generated for the next task cataloged. Workbench is an optional feature of the Operating System that requires extra memory space. The purpose of specifying the WORKBENCH Directive is to reserve addressing space for the Workbench feature of the Operating System. The workbench area is pointed to by TCBKIN in the prologue TCB when that task is active.

SYNTAX

WORKBENCH [REMO[VE]]

[REMO[VE]] - Parameter 1 (optional) indicates that the workbench extension is to be removed for the next task.

EXAMPLES

WORKBENCH Reserve addressing space for the workbench extension.

WORKBENCH REMO Remove the workbench extension.

CHAPTER 4
GUIDE TO THE USE OF TOC/TOC32

4.1 LOGICAL FILES

In common with other MODCOMP software, TOC/TOC32's Input/Output operations are addressed to logical files. TOC/TOC32 reads the binary records of a module to be cataloged from the logical file BI, and outputs binary records onto the logical file BO. TOC/TOC32 examines the logical file CI for directives (if the AO option is selected, it examines the AI file for directives) and outputs to the CO file messages alerting the user of special events. Listings, maps and commands are all output to the LO file.

TOC/TOC32 uses system supported lockout mechanisms whenever a user requests certain functions (CAT, DEL, INI, REM, REN, VAC, COM, RES, SAV, DUP, GET, PUT). These features prevent simultaneous updates to the same load module file by two users, and allow a SQUEEZE on a load module file while programs are loading from it SQUEEZE function locks out each individual load module entry as it is moved. This feature is supported only in Revision 1.0 or later in MAX IV, or Revisions A.0 or later of MAX 32. Refer to the SYSGEN manual for additional information on TOCLOCK.

Figure 4-1 shows the logical files associated with TOC/TOC32 for a user entering directives at a terminal. The user assigns logical file BI to the binary output from an assembler or a link editor, and gives directives to TOC/TOC32 from the terminal to which files CI and CO are assigned. This process can eventually catalog this object onto the load-module file specified. Some TOC/TOC32 Directives (for example, GET, SAVE) generate binary records onto the BO file. If the BO file has been assigned to a magnetic tape, individual modules or entire load-module files can be archived. TOC/TOC32 directives requesting listings (for example, MAP, SOURCE, and LIST) generate these on the LO file, which can be assigned either to a terminal or a printer as desired.

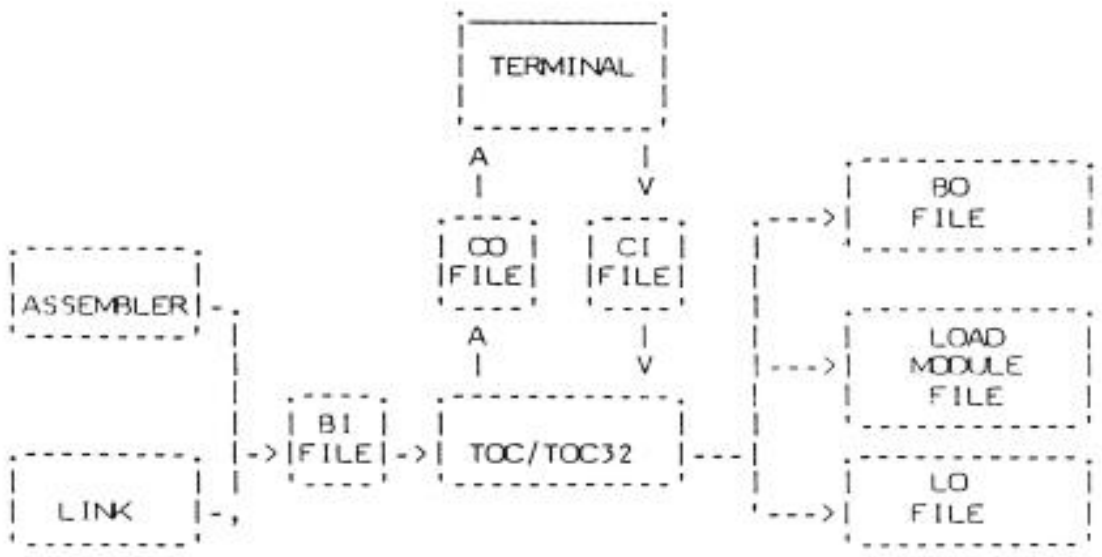


Figure 4-1. Relationship of TOC/TOC32's Logical Files

4.2 TYPES OF LOAD MODULES

TOC/TOC32 creates two types of load modules:

- Task
- Overlay

A task is a program that when activated, has associated with it a unique data structure (the Task Control Block) that permits the task to operate as an independent program in a multiprogramming environment. A task has a priority and is scheduled for execution when all higher priority tasks have suspended their execution. (Example: When tasks are waiting for I/O operations to complete, or when tasks have relinquished CPU time to lower priority tasks, have been placed in a HOLD condition, and so on.)

An overlay is a program that is loaded under the control of an existing task and that is confined to using the Task Control Block of its host task. The host task augments or replaces its own instructions and data with those of the overlay.

TOC/TOC32 generates more information for a task's load module than for an overlay's load module. The information generated for an overlay is a subset of that generated for a task. A program cataloged as a task can be loaded as an overlay, but an overlay can never be activated as a task. When a task is used an an overlay, it takes on the characteristics of an overlay.

4.3 INITIALIZING A LOAD-MODULE FILE

Load-module files must be disc files that have a geometry of one and an unlimited record size. (These terms are defined in the MAX IV or MAX 32 BASIC I/O SYSTEM, System Guide Manual.) Logical files assigned to such disc files can be initialized in the following way:

DIRECTIVES	EXPLANATION
$EXE TOC	
FIL lfn	'lfn' is the logical file name of the load-module file.
INI	Initializes the load module file 'lfn'.
INITIALIZATION OF	lmf requested
/R	Input '/R' to continue or '/A' to abort.
EXI	Exits TOC and returns to Job Control.

The prompt 'INITIALIZATION OF lmf REQUESTED' is generated by TOC/TOC32. Since the initialization of a file destroys all previous information stored on that file, a user should be certain that the user wishes to proceed before responding '/R'.

4.4 CATALOGING AN OVERLAY

The following example is appropriate only if a link-edit is complete and the object is in logical file BO.

DIRECTIVES	EXPLANATION
$EXE TOC	
ASS BI BO	TOC accepts input from the BI file.
REW BI	Positions BI file.
FIL lfn	'lfn' is the load-module file where the overlay is to be cataloged.
OVE	The next cataloged module is an overlay with the name shown in the program (PGM) line.
CAT	Reads the object from BI and catalogs it on 'lfn'.
EXI	Exits from TOC and returns to Job Control.

4.5 CATALOGING A TASK

This example requires a link-edited object module having been written to logical file BO.

DIRECTIVES	EXPLANATION
$EXE TOC	
ASS BI BO	Assigns BI to BO.
REW BI	Rewinds logical file BI.
FIL lfn	'lfn' is the load-module file where the task is to be cataloged.
TAS name 100	The next catalog is to be of a task with the alias 'name' and priority of 100.
PEC REL	The next module is to be relocatable.
LOG 1 LP LP	Logical file 1 for the next module is to be assigned to the Line Printer (LP) device.
LOG 2 TY TY	Logical file 2 for the next module is to be assigned to the terminal device.
LOG 1	Creates space in the next modules TCB for assignment(s) to be made later.
OPT DU	The next cataloged module is to have the Dump Option bit set.
CAT	Reads the object from BI and catalogs it on 'lfn' with the resources specified above.
EXI	Exits from TOC and returns to Job Control.

4.6 ARCHIVING A LOAD-MODULE FILE

The following job stream backs-up a load-module file to magnetic tape media in save format.

DIRECTIVES	EXPLANATION
$EXE TOC	
FIL lfn	'lfn' is the logical file name of the load-module file.
ASS B0 MT1	Assigns binary output to a magnetic-tape-drive device.
REW B0	Rewinds the tape.
SAVE	Copies the file 'lfn' to B0.
ASS BI B0	Assigns binary input to binary output.
REW BI	Rewinds the tape.
TEST	Tests the integrity of the saved modules on tape.
REW BI	Rewinds the tape.
EXI	Exits from TOC and returns to Job Control.

4.7 RESTORING A LOAD-MODULE FILE

The following directive statements take an archived load-module file in save format and restore it to a usable load-module on disc.

DIRECTIVES	EXPLANATION
$EXE TOC	
FIL lfn	'lfn' is the logical file name of the load-module file.
ASS BI MT1	Assigns binary input to magnetic tape drive.
REW BI	Rewinds the logical file BI.
RES	Restores the load-module file.
LIS	Lists the load module file on L0.
EXI	Exits from TOC and returns to Job Control.

4.8 RECATALOGING A LOAD MODULE

The following job stream copies one load-module file to another. The result is that the resources of the module cataloged on 'lm2' are identical to those of the original module on 'lm1'.

DIRECTIVES	EXPLANATION
$EXE TOC	
ASS BI SC BO SC	Assigns BI and BO to the same scratch file (SC).
FIL lm1	Identifies the load-module file 'lm1' on which the desired load module is cataloged.
GET name	Gets the module specified onto SC (by using the BO file).
FIL lm2	Selects the load-module file 'lm2' on which the load module is to be recataloged.
PUT	Catalogs the load module from SC (by using the BI file).
EXI	Exits from TOC and returns to Job Control.

4.9 RECATALOGING A TASK WITH NEW RESOURCES

This job stream recatalogs a load module and modifies the resources of the original load module.

DIRECTIVES	EXPLANATION
$EXE TOC	
ASS BI SC BO SC	Assigns BI to BO to the same scratch file SC
FIL lm1	Selects the load-module file 'lm1' where the task is cataloged.
GET name	Gets the task-load module specified by 'name'.
SOU name INIT	Initiatizes TOC's resource table with the resources of the task 'name'.
FIL lm2	Selects the load-module file where the task with modified resources is to be cataloged.
TAS newname,200	Specifies a 'newname' and priority in TOC's resource table.
PEC PRI REMO	Removes the privileged capability from TOC's resource table.
TMT	Adds a two-map capability to the resource table. (TOC only)
PUT,,NEW	Catalogs the task module (from SC) with the resource table as modified.
EXI	Exits from TOC and returns to Job Control.

4.10 RECATALOGING AND CHANGING THE TCB

The following job stream changes the TCB from one-map to two-map:

```
| DIRECTIVES          EXPLANATION                                    |
|------------------------------------------------------------------|
| $EXE TOC            Execute TOC.                                   |
| ASS BI SC BO SC     Assign BI and BO to scratch file SC.           |
| GET B               Get a copy of batch task B.                    |
| TMTCB               Specify a two-map TCB.  (TOC only)             |
| TASK C,255          Change task name to C with priority 255.       |
| PUT,,NEW            Recatalog with new resources.                  |
```

4.11 JOB CONTROL PROCEDURES

Although an unlimited number of resources can be specified for tasks when using TOC/TOC32, it is better practice (for tasks with numerous resources) to keep resources as part of a job control procedure stored in a user source library. Resources can then be freely modified using the editing facilities of one of the source editors in preference (Examples: SED, MSED, Source Update) to going through the procedure described in Section 4.5 of this chapter.

Editor in preference to going through the procedure described in Section 9 of this chapter.

The identification of '%1', '%2', '%3', '%4', and '%5' are defined by the operator when the procedure is called. Sample Parameter assignments are:

Parameter	Assignment	Description
%1	USR001	Module name
%2	TY	Device assignment
%3	LPP	Device assignment
%4	LM	Load-module file for TSXLDR
%5	LM	Load-module fle to catalog this task (USR001)

The following example shows how standard resources for a user task might be stored in a user source library.

```
$PROC USER %1,%2,%3,%4,%5
$EXE TOC
FIL %5
TASK %1
PECULIAR PRIVILEGED
PECULIAR REMOTE
PECULIAR PIPELINE
IOPERATIONS 4
STACKS 100 12
SYSPAGES 2
ERASE ALL
DEALLOCATE ALL
SPACE 0 ALL
OPTION N00 NO1 NO2 NO3 NO4 NO5 NO6 7
OPTION 8 9 NOA B NOC NOD NOE NOF
LOGFILE TY %2
LOGFILE LP %3
LOGFILE CO TY
LOGFILE CI TY
LOGFILE OC TY
LOGFILE LO TY
LOGFILE AI TY
LOGFILE SCA NO
LOGFILE SCB NO
LOGFILE SC NO
LOGFILE SCC NO
LOGFILE WRK NO
LOGFILE JW NO
LOGFILE UL NO
LOGFILE USL NO
LOGFILE TM NO
LOGFILE SI USL
LOGFILE SO SCB
LOGFILE BI SCA
LOGFILE BO SCA
LOGFILE 1 CO
LOGFILE 2 SI
LOGFILE 3 LO
LOGFILE JC AJC
LOGFILE LB ALB
LOGFILE MC AMC
LOGFILE JM LM
LOGFILES 10
VARIABLES 10
INSERT DUPLICATE TSXLDR %4
CAT
EXI
```

Calling this procedure catalogs the task from the BI file into a specific load-module file.

4.12 CREATING LARGER CONTIGUOUS FREE SPACE

The following steps show how a system manager could consolidate several small non-contiguous holes in the load module file. Suppose that the LIO Directive shows the following:

```
Start       Number      Name
 160          10
 170          10         A
 180          10
 190          10         B
 200          10

Largest hole: 10 at 160
```

The following statements then would condense the space to a largest hole of 30 at 180.

```
LIO                     Get sector usage information
ASS BI SCA BO SCA       File assignments for GET/PUT
REW BI BO               Set at beginning of partition
GET A                   Copy individual modules
GET B
DEL A                   Remove individual modules freeing
DEL B                   their space
PUT                     Recataloging individual modules
PUT
LIO                     Would show new largest hole at 180 for
                        30 sectors.
```

APPENDIX A
QUICK-LOAD FORMAT

The term quick-load format refers to the format in which programs cataloged by TOC/TOC32 are stored on load-module files. The MAX IV/MAX 32 System Loader and TOC/TOC32 use a special-access method to generate and retrieve programs in these load-module files. Only random-access storage devices, capable of transferring large (multisector) records, are suitable mediums for storing these files. These devices must have been specified at system generation time as unlimited record size with a geometry of 1.

Quick-load format consists of the following items:

- An entry in the file's directory that has pointers to...
- One or more Load Program Records (LPRs), which have pointers to...
- One or more Module Object Records (MORs).

The LPRs are themselves a program, interpretable only by the MAX IV/MAX 32 Module Loader, which tells the loader how to load the cataloged application program. The LPRs describe the program's structure and the environment in which it loads. In particular they describe:

- The size of the virtual-addressing space(s) in which the program is loaded.
- The private actual-memory pages to be allocated to the program's addressing space(s) before the program is loaded.
- The shared memory regions (for example, global-common blocks) to be inserted in the addressing space(s), before the program is loaded, and the access rights to be assigned to these regions.
- The manner in which the program is structured on the disc for processing by the loader in the form of pointers to the MORs.
- The manner in which the program is positioned or possibly scattered in its assigned memory-addressing space(s) when loaded, in the form of data chains.

If the program exists as an independent task when loaded, then the LPRs contain additional information. This information defines:

- The intended maximum usage of certain system resources (CPU priority, influence, and so on).

- The size and content of data structures that are constructed by the operating system in order to manage the task.

- System-execution options to be granted to the task, (privileged state, unrollable, and so on).

An overlay does not have this information in its LPRs since it uses the structures already defined by the task that explicitly loads it. The information stored in LPRs is sometimes referred to as the resources, since it described a limited machine environment that must be established by the operating system in order to execute the program.

The MORs of a cataloged program are the body of data and instructions of the program that are to be loaded according to the data chains described in the LPRs. If a large program uses a contiguous block of memory, there is only one MOR, which can be a large multisector disc record. If a program consists of noncontiguous fragments in memory, it has several MORs. Because of MORs, the logical disc devices assigned to load-module files must be capable of incrementally writing multisector records without End-Of-Record (EOR) termination flags. These records are built up sector-by-sector in order to make TOC/TOC32 operate in a limited memory space. Once they are reconstructed, however, they can be read as large multisector, even multitrack, records. Such records can be transferred in one I/O operation by the Load-Module Retriever.

TOC/TOC32 occasionally finds it necessary to break programs into many separate MORs or into single MORs, with readings controlled by several data chain fragments. The following are some reasons why this occurs:

A program can naturally load into several noncontiguous parts. This should be discouraged when load time is of utmost importance.

Some disc devices cannot physically read/write continuous records across a cylinder boundary (usually 2 or 20 tracks).

Some file-management schemes permit disc space to be allocated in noncontiguous units causing MORs to span such gaps occasionally.

The system programmer of a MAX IV/MAX 32 System can impose a size limit for MORs (an integral number of disc revolutions (tracks)), limiting a high-priority program's worst-case access to a disc pack.

Program fragments of less than ten words, if sustained, can exceed maximum transfer rate of the CPU's I/O bus, and are broken into separate I/O transfers that waste the remainder of the current sector.

APPENDIX B
SECURITY FEATURES

TOC/TOC32 operates as an overlay of an existing batch-processing task (generally a low-priority program). Since TOC/TOC32, uncontrolled, can bring almost any kind of task program into being, it is a tool that must be used carefully if system integrity is to be guaranteed. In support of this requirement, the system programmer can install TOC or TOC32 into the operating system in ways that limit their influence on the systems, and limits the power of programs that they can catalog. The following limitations can be imposed:

- TOC can be installed as an unprivileged program, which prevents it from cataloging other programs that are given privileged, unrollable, or other dangerous statuses (TOC only).

- When a load-module file is initialized by TOC/TOC32, that file can be given restrictions that do not permit TOC/TOC32 to catalog programs on it with influence limits or execution priorities above a threshold value.

- When a load-module file is initialized by TOC/TOC32, that file can be restricted to prevent the user from cataloging a program in the file unless a security key is matched to a specified security lock.

APPENDIX C
EXAMPLES OF CATALOGED PROGRAMS

A BATCH-PROCESSING TASK

A batch-processing task can be added to any system as if it were a transient nonresident task. Once such a task becomes active it never exits the system since the process of specifying that it is a batch task causes the system to establish it. If the task program or any overlay calls the EXIT service or is aborted, then the root chain program of the task reloads as an overlay and the privileged/unprivileged state of the original root program restored. Job Control is not the only program that can be cataloged as a root task, but all standard batch-processing tasks require it. In the example below, Job Control is assumed to already be cataloged on global file 'BM' under the alias task name 'B'. The example shows the directives used to catalog a new batch-processing task 'C' by duplicating the task 'B' with unique resources.

EXAMPLE:

```
$JOB- ADD A NEW BATCH PROCESSING TASK "C"
$EXE TOC
ACTION- UNPROTECT SYSTEM DISC FILES
FILE BM
TASK C
PECULIAR PRIV BATCH REMOTE
ERASE ALL
DEALLOCATE ALL
LOGFILE    CI=TY2
LOGFILE    SI=TY2
LOGFILE    CO=TY2
LOGFILE    LO=LP2
LOGFILE    SC=CSC
LOGFILE    SCA=CSA
LOGFILE    SCB=CSB
LOGFILE    WRK=CSW
LOGFILE    JW=CJW
   .
   .
   .
Additional Resources
   .
   .
   .
INS DUP B
CAT
MAP C
LIST
EXIT
```

AN OPERATOR COMMUNICATION DIRECTIVE OVERLAY

An operator directive is processed as a privileged overlay program that loads into MAP 0 in a fixed buffer at a specific virtual address. Such a program must be coded in a special way as described in the MAX IV or MAX 32 OPERATOR COMMUNICATION System Guide Manual. Such a program, assummed to be residing on the first file of a magnetic tape, is cataloged on the global 'SM' file in the following manner. In this example, the overlay is given two duplicate names.

EXAMPLE:

```
$JOB
$ASSIGN BI MT1
$REWIND BI
$ACTION = UNPROTECT "SM" FILE
$EXE TOC
FILE SM
PECULIAR PRIV RELO PIPE
OVERLAY DIE
CAT DIE
DUP DIE LIV
DUP DIE BUR
MAP DIE
LIST
EXIT
$REWIND BI
$NOTE= REMOVE MAG TAPE "MT1"
$ACTION = PROTECT "SM" FILE
```

APPENDIX D
FORMAT OF MAP OUTPUT

The map output lists loader function codes, locations of function codes in the LPRs, and locations of records on the disc.

D.1 SAMPLE TOC MAP OUTPUT

```
#    OPCODE WRDS    MAP OF SIMPLE        DATE  03/20/82
00   SOB 21   1    ONE MAP OBJECT
01   ALO 15   2    ST PAGE= $00  # PAGES=1B    IMAP
02   DMR 17   4    IMAP BIAS=0000 HIGH=1A16    SPACE=0000
03   DMR 17   4    OMAP BIAS=0000 HIGH=1A16    SPACE=0000
04   SOB 21   1    ONE  MAP OBJECT
05   SAC 1D   4     FROM=00C0 TO=00F1  VALUE=0000  IMAP
06   RMO 04 SEC=    310   TA/TC=00F2/0011
07   SAC 1D   4     FROM=0103 TO=01F1  VALUE=0000  IMAP
08   RMO 04 SEC=    311   TA/TC=01F2/1825
09   WAR 0A   4     ADDR=00F0 #/16=0002
0A   WAR 0A   5     ADDR=01F0 #/16=0003
0B   WAR 0A   7     ADDR=0270 #/16=0005
0C   WAR 0A   3     ADDR=02D0 #/16=0001
0D   WAR 0A   4     ADDR=02F0 #/16=0002
0E   WAR 0A   3     ADDR=0320 #/16=0001
0F   WAR 0A   7     ADDR=0340 #/16=0005
10   WAR 0A  65     ADDR=03E0 #/16=003F
11   RLP 05   2    SECTOR=  360
12   WAR 0A  38     ADDR=07D0 #/16=0024
13   WAR 0A  39     ADDR=0B10 #/16=0025
14   WAR 0A  15     ADDR=0D70 #/16=000D
15   WAR 0A  34     ADDR=0E80 #/16=0020
16   RLP 05   2    SECTOR=  361
17   WAR 0A 126     ADDR=1080 #/16=007C
18   RLP 05   2    SECTOR=  362
19   WAR 0A  32     ADDR=1840 #/16=001E
1A   SOB 21   1
1B   NOP 1B   5     I.D.= 1/15/82
1C   DEE 00   2    XFER= 01F6                          ROLLABLE
```

D.2 SAMPLE TOC32 MAP OUTPUT WITH DU OPTION

```
*-13  MODCOMP TASK/OVERLAY CATALOGER  05/07/85

MAP PQUJM DU
     07 MAY 85 10
PQU(0000) A146 9506 0000 0008 0000 0002 9700 0000 /BAT010
PQU(0008) 0000 0000 075B 0000 09FF A142 9402 0000 /BAT010
PQU(0010) 0000 0000 0008 9502 0000 0000 0000 0008 /BAT010
PQU(0018) 8400 003B 2000 FBBD 0000 0000 9D80 0000 /BAT010
PQU(0020) 0443 0000 0460 0000 0000 1B00 8400 0044 /BAT010
PQU(0028) 2000 FFD5 0000 0461 9D00 0000 048C 0000 /BAT010
PQU(0030) 04A4 0000 0000 1B00 8400 0045 2000 FFDF /BAT010
```

```
PQU(0038) 0000 04A5 9D00 0000 04C6 0000 04DE 0000 /BAT010
PQU(0040) 0000 1B00 8400 0046 2000 FF07 0000 04DF /BAT010
PQU(0048) 9D00 0000 05D8 0000 05FF 0000 2020 1B00 /BAT010
PQU(0050) 8400 0048 2000 FFC5 0000 0600 9D00 0000 /BAT010
PQU(0058) 063B 0000 0662 0000 0000 1B00 8400 0049 /BAT010
PQU(0060) 2000 FFE9 0000 0663 9D00 0000 067A 0000 /BAT010
PQU(0068) 06A1 0000 0000 1B00 8400 004A 2000 FF46 /BAT010
PQU(0070) 0000 06A2 8A01 0000 0000 6C00 0500 004C /BAT010
PQU(0078) 4543 5449 5645 0000 2002 5348 4152 494E /BAT010
PQU(0080) 0000 003A /BAT010

*-13  MODCOMP TASK/OVERLAY CATALOGER  05/07/85

#    OPCODE  WRDS          MAP OF PQUJM
00   SOB A1    1     OBJECT FROM LNK32
01   ALO 95    5    ST PAGE= $0008   # PAGES=0002
02   DMR 97    7    BIAS =0000  HIGH=075B    SPACE=02A4
03   SOB A1    1     OBJECT FROM LNK32
04   ERA 94    5    ST PAGE= $0000     # PAGES=0008
05   ALO 95    5    ST PAGE= $0000     # PAGES=0008
06   RMO 84  SEC=#003B    TC/TA=FBBD/0000
07   SAC 9D    7    FROM=#00000443 TO=00000460  VALUE=#00000000
08   NOP 1B    1
09   RMO 84  SEC=#0044       TC/TA=FFD5/0461
0A   SAC 9D    7    FROM=#0000048C TO=000004A4  VALUE=#00000000
0B   NOP 1B    1
0C   RMO 84  SEC=#0045    TC/TA=FFDF/04A5
0D   SAC 9D    7     FROM=#000004C6 TO=#000004DE  VALUE=#00000000
0E   NOP 1B    1
0F   RMO 84  SEC=#0046    TC/TA=FF07/04DF
10   SAC 9D    7     FROM=#000005D8 TO=#000005FF  VALUE=#00002020
11   NOP 1B    1
12   RMO 84  SEC=#0048    TC/TA=FFC5/0600
13   SAC 9D    7     FROM=#0000063B TO=#00000662  VALUE=#00000000
14   NOP 1B    1
15   RMO 84  SEC=#0049    TC/TA=FFE9/0663
16   SAC 9D    7     FROM=#0000067A TO=#000006A1  VALUE=#00000000
17   NOP 1B    1
18   RMO 84  SEC=#004A    TC/TA=FF46/06A2
19   WAR 8A    4     ADDR=00000000 #/16=0001
1A   RLP 05    2     SECTOR=#000004C
      07 MAY 85 10:38:37    /BAT010
PQU(0000) 8A4E 0000 0010 C003 000D B60C 0033 0006 /BAT010
PQU(0008) 1980 C003 0DB6 3300 C30D B630 6186 6300 /BAT010
PQU(0010) DB0C 60CC C6DB 6630 C36D B0DB 1B60 000C /BAT010
PQU(0018) 6330 000D B0C0 0060 030C C00D 006C 0D80 /BAT010
PQU(0020) 0003 0303 0000 1B03 00D8 3630 0006 6C18 /BAT010
PQU(0028) 060C 60C3 3333 1830 30CC 60CC DB60 0180 /BAT010
PQU(0030) C180 0C30 1866 018C 3300 6198 00C0 360C /BAT010
PQU(0038) C18C 0018 0C00 1830 0C06 C0C3 0603 3036 /BAT010
PQU(0040) 0030 6000 C30C 0636 00C0 6C1B 6000 0000 /BAT010
PQU(0048) 036D B6C0 0180 0000 036D B000 6000 0000 /BAT010
PQU(0050) 3000 8A07 0000 06A0 3FFF FFFF FFFF FFFF /BAT010
PQU(0058) FFFF FFFF C000 A101 1B04 3432 3542 484E /BAT010
```

```
PQU(0060) 2020 80D0 0000 0000 9D00 0000 067A 0000 /BAT010
PQU(0068) 06A1 0000 0000 1B00 8400 004A 2000 FF46 /BAT010
PQU(0070) 0000 06A2 8A01 0000 0000 6C00 0500 004C /BAT010
PQU(0078) 4543 5449 5645 0000 2002 5348 4152 494E /BAT010
PQU(0080) 0000 004C /BAT010
1B   WAR 8A   81     ADDR =00000010 #/16=004E
1C   WAR 8A   10     ADDR =000006A0 #/16=0007
1D   SOB A1    1
1E   NOP 1B    5     I.D.= 425BHN
1F   DEE 80    3    XFER=#00000000       PRIVILEGED
```

D.3 COLUMN HEADINGS

The following describes the positioning of items in a line of map output:

 # OPCODE WRDS MAP of xxxxxxx

- '#' is the op code number (hexadecimal position in the LPR) as it appears in the LPR. '#' appears as 'xx' in the following function-code descriptions.
- 'OPCODE' is the mnemonic function code as well as the hexadecimal Loader Function Code.
- 'WRDS' is the number of words the Function Codes takes on disc.
- 'xxxxxx' is the task or overlay name of the map.

D.4 SELECTIONS FOR MAP OUTPUT

NOTE: TOC32 LPR #'s start at hexadecimal position 80, and are displaced by the same values as the standard TOC loader values.

The following are possible items of map output. (Some TOC32 address fields are twice as long as TOC address fields. TOC32 is not rollable.):

xx DEE 00 2 XFER=yyyy PRIVILEGED SYMBIONT ROLLABLE REMOTE UNABORT RELOADROOT

- 'DEE' is define entry point and finish loading.
- 'XFER=yyyy' is the program transfer address in hexadecimal.
- 'PRIVILEGED' indicates the program was cataloged privileged.
- 'SYMBIONT' indicates the program was cataloged as a symbiont.
- 'ROLLABLE' indicates that the task is rollable. (TOC only)
- 'REMOTE' indicates the program is remote.
- 'UNABORT' indicates the task was cataloged unabortable.
- 'RELOADROOT' indicates that the program reloads its root task when it exits.

xx UTI 01 1

- 'UTI' is Use Task Instruction map for loading.

xx UTO 02 1

- 'UTO' is Use Task Operand map for loading.

xx DDL 03 n #ENTRIES=aa

- 'DDL' is Define Directory of Load modules.
- '#ENTRIES=aa' is the number of subprogram maps that follow the main program map.

xx RMO 04 SEC=yyyy TA/TC= aaaa/cccc aaaa/cccc .....

- 'RMO' is Read Module Object records.
- 'SEC=yyyy' is the starting sector (in decimal) to start reading.
- 'TA/TC' stands for Transfer Address (TA) (where to start reading) and Transfer Count (TC) (number of words to read into memory).
- 'aaaa/cccc' is the transfer address and the positive transfer count (in hexadecimal).

xx RLP 05 2 SECTOR=yyyy

- 'RLP' is Read-Loader Program Record.
- 'SECTOR=yyyy' is the disc sector (in decimal) of where to start reading.

xx SSO 06 5 RESET=yyyy SET=zzzz

- 'SSO' is the Set/Reset System Option word.
- 'RESET=yyyy' are the options (in hexadecimal) to reset.
- 'SET=zzzz' are the options (in hexadecimal) to set.

xx SPO 07 5 RESET=yyyy SET=zzzz

- 'SPO' is the Set/Reset Program Option word.
- 'RESET=yyyy' are the options (in hexadecimal) to reset.
- 'SET=zzzz' are the options (in hexadecimal) to set.

xx SNV 08 n NAME=yyy VALUE=zzzz

- 'SNV' is Set Named Variables.
- 'n' is a variable-length op code.
- 'NAME=yyy' is the name of the variable.
- 'VALUE=zzzz' is the value of the variable in hexadecimal.

xx DTV 09 n

- 'DTV' is Delete Task Variables.
- 'n' is a variable-length op code.

xx WAR OA n ADDR=yyyy #/16=zzzz

- 'WAR' is Wait And Relocate module contents (contains the relocation bits if the module is relocatable).
- 'n' is a variable-length op code.
- 'ADDR=yyyy' is the starting address (in hexadecimal) in which to start adding the relocation bias.
- '#/16=zzzz' is the number of words of relocation. Each word relocates 16-loaded words.

xx IMR OB n ADDR=yyyy #/16 = zzzz

- 'IMR' is Immediately Relocate Module contents (do not wait for I/O to complete). The rest of the op code is the same as WAR (OA) above.

```
                                                  [READ    ]
xx MAR OC 2 ST PAGE=[$]yy #PAGES=zz [IMAP] [WRITE   ]
                                    [OMAP] [EXECUTE]
```

- 'MAR' is Modify Access Rights.
- 'ST PAGE=yy' is the Starting Page to start changing the access rights (in hexadecimal).
- '$' indicates that the starting page is relative (load offset is added).
- '#PAGES=zz' is the number of pages (in hexadecimal) that the access rights are to be changed.

- 'IMAP' specifies that the Instruction Map is changed.
- 'READ' specifies that the access rights are changed to read only access.
- 'WRITE' specifies that the access rights are changed to read/execute/write (no restrictions) access.
- 'OMAP' specifies that the Operand Map is changed.
- 'EXECUTE' specifies that the access rights are changed to read/execute access.

```
                [READ    ]
xx IGS OD 7     [EXECUTE] name [$] yyyy [IMAP]
                [WRITE   ]                [OMAP]
```

- 'IGS' is Insert Globally-defined Shared area.
- 'READ' is read access rights.
- 'EXECUTE' is read/execute access rights.
- 'WRITE' is read/execute/write access rights.
- 'name' is the 1- to 6-byte name of the globally-defined shared area.
- '$' indicates that the next address is relative.
- 'yyyy' is the address in the map (in hexadecimal) where the area is to be inserted.
- 'IMAP' indicates that the shared area is inserted in the Instruction Map.
- 'OMAP' indicates that the shared area is inserted in the Operand Map.

```
                      [IMAP]
xx EGS OE 4 name      [OMAP]
```

- 'EGS' is Extract Globally-defined Shared area.
- 'name' is the 1- to 6-byte name of the globally-defined shared area.
- 'IMAP' indicates the area is removed from the Instruction Map.
- 'OMAP' indicates the area is removed from the Operand Map.

```
xx PSC OF 7
```

- 'PSC' is Create Private-Shared region.

```
xx IPS 10 7
```

- 'IPS' is Insert Private-Shared region.

```
xx EPS 11 4
```

- 'EPS' is Extract Private-Shared region.

```
xx ISM 12 7 yyyy zzzz wwww aaaa nnnnnn ffff
```

- 'ISM' is an Insert-Shared Load Module.
- 'yyyy' is #1200 followed by the option bits.
- 'zzzz' is the read key in hexadecimal.
- 'wwww' is the write key in hexadecimal.
- 'nnnnnn' is the name of the module.
- 'aaaa' is the address of the insert.
- 'ffff' is the logical file name.

```
xx ESM 13 3
```

- 'ESM' is Extract Shared-load Module.

```
xx ERA 14 2    [ALL                                ]  [IMAP]
               [ST PAGE = [$]yy  #PAGES = zz]  [OMAP]
```

- 'ERA' is Erase All shared-page regions.
- 'ALL' specifies erase all shared pages starting at page 0 and ending at page #FF.
- 'ST PAGE=yy' is the Starting Page of the erase (in hexadecimal).
- '$' specifies that the following page is relative (load offset is added).

- '# PAGES=zz' is the number of pages to be erased (in hexadecimal).
- 'IMAP' indicates the erase is done in the Instruction Map.
- 'OMAP' indicates the erase is done in the Operand Map.

```
xx ALO 15 2  [ALL                          ]    [IMAP]
             [ST PAGE=[$]yy #PAGES=zz]      [OMAP]
```

- 'ALO' is Allocate private pages.
- 'ALL' specifies that the entire map is allocated.
- 'ST PAGE=yy' is the Starting Page of the allocate (in hexadecimal).
- '$' indicates that the next page is relative (load offset is added).
- '#PAGES=zz' is the number of pages to be allocated (in hexadecimal).
- 'IMAP' indicates the allocate is done in the Instruction map.
- 'OMAP' indicates the allocate is done in the Operand map.

```
xx DEA 16 2  [ALL                              ]  [IMAP]
             [ST PAGE=[$]yy #PAGES = zz]       [OMAP]
```

- 'DEA' is Deallocate private pages. The rest of this op code is the same as ERA (#14) except that private pages, instead of shared pages, are to be deallocated.

```
xx DMR 17 4  [IMAP] BIAS=yyyy  HIGH=zzzz SPACE=aaaa
             [OMAP]
```

- 'DMR' is Define Module's virtual Ranges.
- 'IMAP' indicates the defining of the Instruction Map.
- 'OMAP' indicates the defining of the Operand Map.
- 'BIAS=yyyy' is the bias (in hexadecimal) from the BIAS Directive at catalog time.
- 'HIGH=zzzz' is actual high address (in hexadecimal) of the program when it was cataloged.
- 'SPACE=aaaa' is the amount of extra load-time space (in hexadecimal) from the first parameter of the SPACE Directive when the program was cataloged.

```
xx FAR 19 4 LOGFILE lfn,now,def
```

- 'FAR' is define the contents of one temporary 'FAT' (File Assignment Table). 'FAR' can be cleared by 'CTA' (#1F).
- 'lfn' is the logical file name.
- 'now' is the current assignment.
- 'def' is the default assignment.

```
xx NOP 1B 5
```

- 'NOP' is a variable-length no-operation op code. This op code contains the date or ID when the program was cataloged.

```
xx SLO 1C 1
```

- 'SLO' is Set Loading Options (options to the load-module loader).

```
xx SAC 1D 4  FROM=yyyy  TO=zzzz  VALUE=cccc  [IMAP]
                                             [OMAP]
```

- 'SAC' is Set Area Constant (a RES x,y from the assembler).
- 'FROM=yyyy' is the starting address (in hexadecimal) in which to start storing the constant.
- 'TO=zzzz' is the ending address (in hexadecimal) in which to store the constant.
- 'VALUE=cccc' is the constant (in hexadecimal) that is to be stored.
- 'IMAP' indicates instruction-space area.
- 'OMAP' indicates operand-space area.

xx ICM 1E 5 yyyy aaaa ffff nnnnnn

- 'ICM' is an insert copiable module.
- 'yyyy' is #1E00 followed by the option bits.
- 'aaaa' is the address of the load.
- 'ffff' is the logical-file name.
- 'nnnnnn' is the name of the module.

xx CTA 1F 1

- 'CTA' is clear all temporary FATs (clears all FAR (#19) and FAD (#38) op codes).

xx SOB 21 1 [PIPELINE/][[ONE]MAP OBJECT [FROM M4EDIT]][/RIOCOMPLETE] [TWO]

- 'SOB' is Start/Stop Object. 'SOB' defines the Start of Object (such as RMO and SAC) and also the end of object. When SOB defines the start of object, the descriptors below can appear:

  - 'PIPELINE' indicates the program was cataloged with the Directive statement PECULIAR PIPELINE.
  - 'ONE' indicates that program was link edited to execute in one map.
  - 'TWO' indicates that program was link edited to execute in two maps.
  - 'FROM M4EDIT' indicates that the MAX IV link editor was used in link editing the task/overlay.
  - 'RESUME I/O' indicates the program was cataloged with peculiar RIOCOMPLETE to be resumed if suspended when an I/O operation completes.

xx UDC 22 n

- 'UDC' is the User-Defined op Code.
- 'n' is the variable length of this op code.

xx SPD 30 B [ONEMAP TASK] TCB PAGES=yy IMAP PAGES=zzz OMAP PAGES=yyy [TWOMAP TASK]

STACK=pppp/rrrr [AUTOSTART] PRIORITY=zz LIMIT=yy
PRIVATE NODES=zz GLOBAL NODES=yy

- 'SPD' is Set Prologue TCB Definitions. The appearance of this op code defines a ask and is always the first op code generated. Overlays never use this op code.
- 'ONEMAP TASK' indicates this is a single-map task.
- 'TWOMAP TASK' indicates this is a two-map task.
- 'TCB PAGES = yy' is the number of memory pages (in hexadecimal) required to hold this task's TCB.
- 'IMAP PAGES = zzz' is the number of pages (in hexadecimal) that the instruction map holds (#1 to #100).
- 'OMAP PAGES = yyy' is the number of pages (in hexadecimal) that the operand map holds (#1 to #100).
- 'STACK pppp/rrrr' specifies the stack sizes. 'pppp' is the main push-stack size and 'rrrr' is the PS/PR stack size.
- 'AUTOSTART', if present, specifies that the task was cataloged as auto-start. The task does not auto-start unless the task is pre-queued.
- 'PRIORITY=zz' is the run-time priority cataloged with the task.
- 'LIMIT=yy' is the limit priority cataloged with the task.

- 'PRIVATE NODES=zz' is the number of private nodes -1 (in hexadecimal) that is allocated when the task starts.
- 'GLOBAL NODES=yy' is the counter of the maximum number of global nodes the task can use simultaneously.

xx DPS 32 1 #ENTRIES=aa

- 'DPS' is Define Page-Sharing extension.
- '#ENTRIES=aa' is the number of entries (in hexadecimal) the extension can hold. The number of words in this extension is (aa*7)+1.

xx DCA 33 1 #ENTRIES=aa

- 'DCA' is define compatible-allocation extension (MAX III memory allocation).
- '#ENTRIES=aa' is the maximum number of active allocations that the task can perform. The number of words in this extension is aa+2.

xx DVE 34 1 #ENTRIES=aa

- 'DVE' is Define Variables Extension.
- '#ENTRIES=aa' is the number of vacant variables (in hexadecimal) the extension can hold. The number of words in this extension is (aa*2)+2.

xx DUC 36 3 LIMIT COUNT = yyyyyyyy

- 'DUC' is Define-task Utilization limit/Count.
- 'LIMIT COUNT = yyyyyyyy' is the 32-bit limit count (in hexadecimal).

xx FAV 37 1 #ENTRIES=yy

- 'FAV' is define 'n' number of vacant file assignments.
- '#ENTRIES=yy' is the number of vacant FAT's (in hexadecimal).

xx FAD 39 4 LOGFILE lfn,now,def

- 'FAD' is define contents of one permanent FAT (file assignment). This op code cannot be cleared by CTA (#1F).
- 'lfn' is the logical file name.
- 'now' is the current assignment.
- 'def' is the default assignment.

xx DFL 3B 1 #ENTRIES=aa

- 'DFL' is Define FAT List length.
- '#ENTRIES=aa' is the total number of all file assignments (in hexadecimal), including temporary, permanent, and vacant. The number of words in this list is aa+2.

xx SPL 3C 1 #ENTRIES=aa

- 'SPL' is define System-Page Length and defines the number of Map 0 pages that can be used by services.
- '#ENTRIES=aa' is the number of pages that can be used (in hexadecimal). The number of words in this list is aa+1.

xx USE 3D 1 #ENTRIES=yy

- '#ENTRIES=yy' is the number of words in the user's extension (in hexadecimal).

xx WKB 3E 1 WORKBENCH

- 'WKB' is the Workbench Extension.

xx CLE 3F 1 Command Line Extension

. 'CLE' is the Command Line Extension.

APPENDIX E
SAMPLE TOC/TOC32 LISTINGS

Listing output for TOC and TOC32 looks the same.

E.1 SAMPLE LDI DISPLAY

```
*LD MAX IV LOAD MODULE FILE TM     DATE       6/15/85

ENTRY   NAME       SEC   #SEC       I.D.       DUPLICATE

 1      FORMS        3
 2      LDMEM        5
 3      MEM          7
 4      SUB          9
 5      TASS        11
 6      FMG         13
 7      CTR         15
 8      SP          17
 9      SP5         30
10      DSP         43
11      MMM         51
12      SP9         53
13      LIST        66
14      QM          68
15      COMW        90
16      COMR        94
17      TEST        98
18      TEST1      100
19      PQU        103
20      FMTEST     109
21      LOAN       111
22      COMWTT     175
23      NWT        179
24      XXXX       183
25      LOAN3      185
26      LOANX      249
27      LOANXX     314
28      TEST12     379
29      TEST13     381
30      TSTCTR     383
31      DINT       387
32      DUMB       440
33      PASCAL     442
34      TSTCT2     458

NUMBER OF ENTRYS AVAILABLE IN DIR = 62
NUMBER OF ENTRYS USED IN DIR = 34
NUMBER OF SECTORS AVAILABLE IN FILE = 640
NUMBER OF SECTORS USED FOR DIRECTORY = 2
NUMBER OF SECTORS USED IN DUPLICATE DIR = 1
NEXT AVAILABLE SECTOR = 462
```

E.2 SAMPLE LIS DISPLAY (PARAMETER 2 NOT SPECIFIED):

```
*LO MAX IV LOAD MODULE FILE TM                    DATE 6/15/85

ENTRY  NAME    SEC     #SEC     I.D.          DUPLICATE

  1    FORMS     3       2    04/01/80                    OVERLAY
  2    LDMEM     5       2    03/06/80                    OVERLAY
  3    MEM       7       2    03/06/80                    OVERLAY
  4    SUB       9       2    03/06/80                    OVERLAY
  5    TASS     11       2    03/07/80                    OVERLAY
  6    FMG      13       2    03/07/80                    OVERLAY
  7    CTR      15       2    03/07/80                    OVERLAY
  8    SP       17      13    03/19/80                    OVERLAY
  9    SP5      30      13    04/16/80                    OVERLAY
 10    DSP      43       8    05/24/80                    OVERLAY
 11    MMM      51       2    04/01/80                    OVERLAY
 12    SP9      53      13    04/03/80                    OVERLAY
 13    LIST     66       2    05/08/80                    OVERLAY
 14    QM       68      22    05/28/80                    OVERLAY
 15    COMW     90       4    05/12/80                    OVERLAY
 16    COMR     94       4    05/12/80                    OVERLAY
 17    TEST     98       2    05/16/80                    OVERLAY
 18    TEST1   100       3    05/27/80                    OVERLAY
 19    PQU     103       6    06/02/80                    OVERLAY
 20    FMTEST  109       2    06/27/80                    OVERLAY
 21    LOAN    111      64    07/22/80                    OVERLAY
 22    COMWTT  175       4    07/25/80                    OVERLAY
 23    NWT     179       4    07/30/80                    OVERLAY
 24    XXXX    183       2    09/03/80                    OVERLAY
 25    LOAN3   185      64    09/04/80                    OVERLAY
 26    LOANX   249      65    11/18/80                    OVERLAY
 27    LOANXX  314      65    12/09/80                    OVERLAY
 28    TEST12  379       2    01/22/81                    OVERLAY
 29    TEST13  381       2    01/22/81                    OVERLAY
 30    TSTCTR  383       4    06/23/81                    OVERLAY
 31    DINT    387      53    04/23/81                    OVERLAY
 32    DUMB    440       2    06/02/81                    OVERLAY
 33    PASCAL  442      16    06/22/81                    OVERLAY
 34    TSTCT2  458       4    06/23/81                    OVERLAY

NUMBER OF ENTRYS AVAILABLE IN DIR = 24
NUMBER OF ENTRYS USED IN DIR = 34
NUMBER OF SECTORS AVAILABLE IN FILE = 640
NUMBER OF SECTORS USED IN FILE = 462
NUMBER OF SECTORS USED FOR DIRECTORY = 2
NUMBER OF SECTORS USED IN DUPLICATE DIR = 1
NEXT AVAILABLE SECTOR = 462
```

E.3 SAMPLE LAE DISPLAY

Assume the following directory for a LIST,

```
*L.0 MAX IV LOAD MODULE FILE Y                    DATE 6/15/85

ENTRY  NAME    SEC      #SEC     I.D.          DUPLICATE

 1     DINT      2       55    02/08/82                      OVERLAY
 2     SIMPLE   57       55    04/07/84                      OVERLAY
 3     FOR     112       54    04/19/84                      OVERLAY
 4     S       166        4    04/25/84      SIMPLE          OVERLAY
 6     TOCTES  421        5    04/15/84                      OVERLAY
 7     OVTOC   426       31    04/21/84                      OVERLAY
 8     TOCX    457       89    04/25/84                      OVERLAY
NUMBER OF ENTRYS AVAILABLE IN DIR =      31
NUMBER OF ENTRYS USED IN DIR =        7
NUMBER OF SECTORS AVAILABLE IN FILE =    640
NUMBER OF SECTORS USED IN FILE =     295
NUMBER OF SECTORS USED FOR DIRECTORY =      1
NUMBER OF SECTORS USED IN DUPLICATE DIR =       1
HIGH AVAILABLE SECTOR =      546
LARGEST NUMBER CONTIGUOUS UNUSED SECTORS =    251
FILE NEEDS TO BE COMPRESSED
```

---

then the LAE S+++++ would generate

```
*L.0 MAX IV LOAD MODULE FILE Y                    DATE 6/15/85

ENTRY  NAME    SEC      #SEC     I.D.          DUPLICATE
 2     SIMPLE   57       55    04/07/84                      OVERLAY
 4     S       166        4    04/25/84      SIMPLE          OVERLAY
```

---

and the LAE +O++++ would generate

```
*L.0 MAX IV LOAD MODULE FILE Y                    DATE 6/15/85

ENTRY  NAME    SEC      #SEC     I.D.          DUPLICATE
 3     FOR     112       54    04/19/84                      OVERLAY
 6     TOCTES  421        5    04/15/84                      OVERLAY
 8     TOCX    457       89    04/25/84                      OVERLAY
```

E.4 SAMPLE LIO DISPLAY

Assume the following LIST directory:

```
*L.O MAX IV LOAD MODULE FILE Y                    DATE 6/15/85

ENTRY NAME    SEC      #SEC     I.D.          DUPLICATE

 1    DINT      2       55    02/08/82                      OVERLAY
 2    SIMPLE   57       55    04/07/84                      OVERLAY
 3    FOR     112       54    04/19/84                      OVERLAY
 4    S       166        4    04/25/84        SIMPLE        OVERLAY
 6    TOCTES  421        5    04/15/84                      OVERLAY
 7    OVTOC   426       31    04/21/84                      OVERLAY
 8    TOCX    457       89    04/25/84                      OVERLAY
NUMBER OF ENTRYS AVAILABLE IN DIR =      31
NUMBER OF ENTRYS USED IN DIR =        7
NUMBER OF SECTORS AVAILABLE IN FILE =    640
NUMBER OF SECTORS USED IN FILE =     295
NUMBER OF SECTORS USED FOR DIRECTORY =      1
NUMBER OF SECTORS USED IN DUPLICATE DIR =       1
HIGH AVAILABLE SECTOR =      546
LARGEST NUMBER CONTIGUOUS UNUSED SECTORS =   251
FILE NEEDS TO BE COMPRESSED
```

---

then LIO would generate:

```
*L.O MAX IV LOAD MODULE FILE Y                    DATE 6/15/85

      START      NUM        NAME

          0        1        DIRECTORY
          1        1        DIRECTORY
          2       55        DINT
         57       55        SIMPLE
        112       54        FOR
        166        4        S
        170      251
        421        5        TOCTES
        426       31        OVTOC
        457       89        TOCX
        546       94

LARGEST HOLE:       251 at 170

START - Start sector
NUM   - Number of sectors (module size)
NAME  - Module name with the Keyword DIRECTORY indicating a
directory or duplicate directory sector.
```

NOTE: If NUM (number of sectors for this module) is zero, then the module is greater than 1023 sectors in size. The value of zero implies that all space up to the next module is considered used and therefore not available.

Complex duplicates cataloged with versions of TOC prior to Revision K.0, will show a zero number of sectors in the LIO listing. To obtain the correct size, and free the space for cataloging other modules, recatalog each complex duplicate.

Segmented programs cataloged with versions of TOC prior to Revision K.0, will show an inaccurate number of sectors in the LIO listing.

This value erroneously indicates free space that can be used in cataloging activities.

To correct this, the user must do one of the following before cataloging any new programs with Revision K.0 or subsequent revision of TOC:

- Delete all segmented programs and then recatalog them with revision K.0 or subsequent revision of TOC.
- GET all programs and then PUT all programs under either revision of TOC.
- Compress the load module file under either revision of TOC.

APPENDIX F
ERROR MESSAGES

Error messages from TOC/TOC32 are written to the system global file CO, which is normally assigned to the console device. The operator's response is read from the CI file.

Assignment of CI:

- If CI is assigned to TY and an error occurs, the appropriate error message is printed on TY (by the REX MESSAGE service). The system remains in TOC/TOC32 and waits for a new directive.

- If CI is not assigned to TY and:

  - The mode is Verify, when an error occurs a REX ABORT takes place with the reason code @TOC.

  - The mode is Noverify, when an error occurs a REX MESSAGE with HOLD option takes place. In this case, a '/R' (RESUME) or '/A' (ABORT) must be entered. A '/A' always aborts TOC/TOC32. A '/R' continues execution of the current directive. If the directive cannot be continued, the HOLD state is reentered.

The following is an explanation of TOC/TOC32's error messages, in alphabetical order ('xxx' is a variable for a directive).

xxx IS IN DIRECTORY

Commands Used: CATALOG, DUPLICATE, or PUT

Meaning: The file name 'xxx' is already in the directory.

Recommended Action: Enter '/R' to overwrite the current entry with the new entry or enter '/A' to abort.

16-BIT FORMAT BINARY NOT ALLOWED WITH TOC32

Command Used: CAT

Meaning: 16-bit binary records have been read from BI.

Recommended Action: Check assignment of BI, or use TOC.

BIAS MUST BE SPECIFIED WITH LNK32

Command Used: BIAS

Meaning: Bias command was entered under TOC32.

Recommended Action: Need to bias program with LNK32 prior to catalog.

BIAS NOT ALLOWED WITH M4EDIT

Commands Used: CATALOG

Meaning: The task/overlay cataloger BIAS Directive cannot be processed when cataloging/recataloging programs that have been link edited by the MAX IV or MAX 32 Link Loader.

Recommended Action: Either a bias of 0 must be entered before cataloging, or the program must be link-edited again with a bias.

BUFFER FULL

Commands Used: LOGFILE, REASSIGN, TIMELIMIT or VARIABLE

Meaning: TOC/TOC32's internal temporary buffer is full. It holds logfiles.

Recommended Action: Decrease the number of resources.

CHECKSUM ERROR

Commands Used: PUT, RESTORE, or TEST

Meaning: TOC/TOC32 has read a standard binary record with an erroneous checksum.

Recommended Action: Check assignment and positioning of the BI file. Use backup file to recover.

COMPLEX DUPLICATE xxx HAS BEEN DELETED

Commands Used: CATALOG or PUT

Meaning: A module that had complex duplicates was cataloged or put, so all complex duplicates of that module have been deleted.

Recommended Action: Recreate the complex duplicates.

DATE NOT ENTERED

Commands Used: TOC or PROCESSOR

Meaning: Verify Mode Only: This message allows the operator a chance to enter the date.

Recommended Action: Enter the date (use OC directive //DATE) and then enter '/R' to continue or enter '/A' to abort.

DELETE OF xxx REQUESTED

Command Used: DELETE

Meaning: Verify Mode Only: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

DIRECTORY NOT INITIALIZED

Commands Used: CATALOG, COMPRESS, LDIRECTORY, LIST, MAP, PUT, SAVE or SUBSTITUTE

Meaning: The logical file is not initialized. Either the file name was entered incorrectly with the FILE Directive or the file was not initialized.

Recommended Action: Reenter the FILE Directive using the correct file name.

END OF MEDIA

Commands Used: CATALOG, DUPLICATE, MAP, RENAME, RESTORE, SAVE, SOURCE, SUBSTITUTE or VACATE.

Meaning: A record could not be written on a storage device, such as a disc partition, because that record would have fallen outside the limits of the storage medium.

Recommended Action: If a load module file is involved, a COMPRESS can result in more room becoming available.

ERROR! RELOCATABLE PROGRAM HAS EXCEEDED MAXIMUM SIZE

Commands Used: CAT

Meaning: Peculiar relocatable has been specified and the program is larger than #80000 16-bit words in size.

Recommended Action: 1. Reduce size. 2. Don't specify peculiar relocatable.

FILE LOCK ERROR, STATUS = #000X

Commands Used: CATALOG, DELETE, INITIALIZE, REMOVE, RENAME, VACATE, COMPRESS, RESTORE, SAVE, DUPLICATE, GET, PUT or SQUEEZE

Meaning: The requested load module file is being modified by another batch task; or there is no available map zero data structure.

Recommended Action: Try later after other user finishes; determine if "TOCLOCK" was specified at SYSGEN, the status is the condition codes from the Processor REX Service.

ILLEGAL BINARY RECORD

Commands Used: PUT, RESTORE, or TEST

Meaning: TOC has read a standard-binary record and the first byte is not #03 or #07.

Recommended Action: Check positioning and assignment of the BI file.

ILLEGAL DIRECTIVE

Commands Used: TOC or PROCESSOR

Meaning: The directive entered cannot be found in the table of legal commands in TOC/TOC32.

Recommended Action: A legal directive must be entered.

ILLEGAL DUPLICATE FORMAT

Commands Used: COMPRESS, LDIRECTORY, LIST, or SAVE

Meaning: A program that is being pointed to by a duplicate name is not in the load-module directory.

Recommended Action: Recovery is not possible.

ILLEGAL FILE/DEVICE NAME

* **Commands Used:** AVFILE, AVRECORD, BKFILE, BKRECORD, COMPRESS, MAP, RENAME, REWIND, SAVE, SOURCE, SUBSTITUTE, VACATE or WEOF

  Meaning: The file name is illegal. For further information refer to the discussion of the REX TASS Service (Register 3 Bit 2 set indicates offline - illegal) in the MAX IV or MAX 32 Executive (REX) Services System Guide Manual.

  Recommended Action: Reenter the directive with correct file name.

* **Commands Used:** ASSIGN, FILE or RESTORE

  Meaning: The file name is illegal. For further information refer to the REX ASSIGN service in the MAX IV or MAX 32 Executive (REX) Services System Guide Manual.

  Recommended Action: Reenter the directive with the correct file name.

* **Commands Used:** COPY or GET

  Meaning: The BI or BO file is not assigned correctly.

  Recommended Action: Check assignment and positioning of BI and BO file.

ILLEGAL FUNCTION CODE

Commands Used: DUPLICATE

Meaning: Wrong sequence of quick-load op codes. Either:
- the first is DEE or RMO,
- the second is DEE, or
- the third is RMO.

Recommended Action: Recovery is not possible.

ILLEGAL RESOURCE FOR AN OVERLAY

Commands Used: IOPERATIONS, PAGESHARE, PECULIAR or WORKBENCH

Meaning: The OVERLAY Directive was entered prior to one of these directives:

- IOPERATIONS
- PAGESHARE
- WORKBENCH

OR the OVERLAY Directive was entered prior to one of these parameters of the PECULIAR Directive:

- Symbiont
- Unrollable
- Remote
- Autostart
- Unabortable
- Batch

Recommended Action: Catalog the overlay without entering the directive defining the illegal resource or modify the resource so it is legal for an overlay.

ILLEGAL SAVE FORMAT

Commands Used: PUT, RESTORE

Meaning: The program or load-module file is not in the proper format that TOC/TOC32 uses for copying/saving.

Recommended Action: Check assignment and positioning of the BI file.

ILLEGAL SECURITY KEY

Commands Used: ASSIGN, COMPRESS, FILE, LDIRECTORY, LIST or RESTORE

Meaning: The load-module file being used contains a security key name that does not match the one entered in the KEY Directive. (Refer to the INITIALIZE or SUBSTITUTE Directives for further information.)

Recommended Action: Reenter the KEY Directive with the right security-key name.

INITIALIZATION OF xxx REQUESTED

Commands Used: INITIALIZE

Meaning: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

LARGEST HOLE SIZE ZERO

Commands Used: LIO, LAE, CAT or PUT

Meaning: The largest free space.

Recommended Action: Remove or copy out and remove module entries to free up space. It is also possible that the last module on the file is larger than 1023 sector which shows up as zero.

LIMIT PRIORITY NOT HIGH ENOUGH

Commands Used: ASSIGN, FILE, RESTORE

Meaning: The priority level of the load module is higher in priority than TOC/TOC32.

Recommended Action: Execute a version of TOC/TOC32 with a higher-limit level or lower the level of the file.

MINIMUM CATALOGER LEVEL BEING RAISED

Commands Used: SUBSTITUTE

Meaning: Verify Mode Only: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

MODIFY AT n VALUE n NOT PROCESSED

Commands Used: MODIFY

Meaning: The modify address could not be found within a RMO. Set Area Constant (SAC) load codes can not be modified.

Recommended Action: Verify input.

MODIFY BUFFER FULL

Commands Used: MODIFY

Meaning: TOC's internal buffer is full.

Recommended Action: If more modifies are needed, CATALOG, then do a GET, enter more modifies, and then do a PUT.

MODULE NOT FOUND

Commands Used: DELETE, DUPLICATE, GET, MAP, RENAME, or SOURCE

Meaning: The name of the module was not found in the directory.

Recommended Action: Verify the name of the module and reenter the directive with the correct name.

MODULE REQUIRES LINK EDITING

Command Used: CATALOG

Meaning: A function code that TOC/TOC32 cannot process was detected (location counter, external reference, common origin, or common reference).

Recommended Action: Link edit the module and then execute TOC/TOC32.

MODULE xxxxxx LOCKED, STATUS = #000X

Commands Used: SQUEEZE

Meaning: The named module has been locked by the module loader (it is either a segmented module or a shared module).

Recommended Action: SQUEEZE tries five times to lock the module with a delay in between each lock request. SQUEEZE terminates the squeeze process when this message is printed with the load module partially processed. The status is the condition codes from the Processor REX Service.

MODULE xxxxxx RELOCATED

Commands Used: SQUEEZE

Meaning: This is for information only. This message identifies the fact that a module has been successfully moved and recataloged on the load module file.

Recommended Action: No response is required.

MOUNT NEXT TAPE

Commands Used: CATALOG, RENAME or TEST

Meaning: The BI file has been searched to and End-Of-File or End-Of-Media and the module was not found.

Recommended Action: Check the assignment of the BI file and enter /R to resume or /A to abort.

NEW NAME IN DIRECTORY

Commands Used: RENAME

Meaning: The new name is already in the directory.

Recommended Action: Use a different name or remove the name from the directory.

NO DUPLICATE MASTER CATALOGED

Commands Used: SAVE or COMPRESS

Meaning: TOC/TOC32 searched to the last segment of the sector and the duplicate master was not found.

Recommended Action: Either the duplicate must be removed or the master must be recataloged.

NOT ASSIGNED TO A DISC

Commands Used: ASSIGN, COMPRESS, FILE, LDIRECTORY, LIST or RESTORE

Meaning: The file name is not assigned to a disc partition. (Refer to discussion of REX TASS service in the MAX IV or MAX 32 Executive (REX) Services System Guide Manual.

Recommended Action: The file name must be reassigned to a disc partition.

NOVERIFY WANTED

Commands Used: INITIALIZE or SUBSTITUTE

Meaning: This message allows the operator a chance to reconsider switching to Noverify Mode.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

OVTOC NOT CATALOGED RELOCATABLE
OVT32 NOT CATALOGED RELOCATABLE

Commands Used: TOC or PROCESSOR

Meaning: The directive overlay for TOC/TOC32 has not been cataloged as a relocatable overlay.

Recommended Action: Recatalog OVTOC as relocatable.

OVTOC NOT ON SAME LOAD MODULE AS TOC
OVT32 NOT ON SAME LOAD MODULE AS TOC32

Commands Used: TOC or PROCESSOR

Meaning: The directive requested was not found or TOC/TOC32 could not find the overlay OVTOC or OVT32.

Recommended Action: Enter a legal directive or change OVTOC or OVT32 to be an overlay on the same load-module file as TOC/TOC32.

PAGE SHARING EXTENSION TOO SMALL

Commands Used: CATALOG

Meaning: The size of the page sharing extension (defined by the PAGESHARE Directive) is less than the size of the page shares needed (determined by the INSERT Directive).

Recommended Action: Increase size of pageshare extension.

PGM PLUS SPACE TOO LARGE (TOC only)

Commands Used: CATALOG or DUPLICATE

Meaning: A partial IMAP or OMAP is wanted and the number of pages needed for the TCB is greater than the total size of the area. The maximum area is #FF pages.

Recommended Action: Respecify the ISPACE or OSPACE Directive with a smaller value.

PRIVILEGED STATE WANTED

Commands Used: PECULIAR

Meaning: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

RELOCATING MODULE xxxxxx

Commands Used: SQUEEZE

Meaning: This is for information only. This identifies which module SQUEEZE is currently operating on and if a system failure occurs, this module can be found on the scratch file or still in the load module file.

Recommended Action: No response is required.

RENAME OF MODULE WANTED

Commands Used: RENAME

Meaning: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

RESTORE REQUESTED

Commands Used: RESTORE

Meaning: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

SAVE COMPLETED

Commands Used: COMPRESS

Meaning: This is an informational message that informs the operator that the save portion of the COMPRESS Directive is complete.

Recommended Action: No response is required.

SEQUENCE ERROR

Commands Used: PUT, RESTORE or TEST

Meaning: TOC/TOC32 has read a standard binary record with a wrong sequence number.

Recommended Action: Check assignment and positioning of BI file.

STATEMENT ERROR

Commands Used: (Any directive)

Meaning: The directive statement does not conform to the syntax rules.

Recommended Action: Verify the syntax of the last directive entered (refer to Chapter 3) and reenter directive correctly.

SQUEEZE NOT SUPPORTED ON THIS DISC TYPE

Commands Used: SQUEEZE

Meaning: Older disc types that do not support cylinder boundary overflow are not supported on the squeeze function.

Recommended Action: Move load module to supported disc type.

TAPE LOW

Commands Used: COPY, GET or SAVE

Meaning: An End-Of-Media was encountered while doing output to paper tape.

Recommended Action: More paper tape is required in the paper tape punch to continue.

TCB TOO LARGE

Commands Used: CATALOG, DUPLICATE or PUT

Meaning: The size of the TCB needed for the current task exceeds the maximum page limit.

Recommended Action: Check resources and reenter directive. Some resources may have to be deleted or removed if cataloging.

TOC NOT PRIVILEGED

Commands Used: PECULIAR, PUT, RESTORE or SAVE

Meaning: TOC/TOC32 is being executed as an unprivileged program and a privileged directive or a directive involving a privileged module or program was executed.

Recommended Action: Execute a privileged version of TOC/TOC32.

TOO MANY SUBPROGRAMS

Commands Used: PUT or RESTORE

Meaning: The allowable number of segment subprograms has been exceeded.

Recommended Action: Reduce the number of subprograms.

UNIDENTIFIED QUICK LOAD OP CODE

Commands Used: CATALOG, DUPLICATE, MAP, PUT, RESTORE, SAVE, SOURCE, TIMELIMIT, or VARIABLE

Meaning: The quick-load op code is not legal in TOC/TOC32's op code table (or is SPD or IGNORE).

Recommended Action: Check assignment and positioning of BI file. The allowable limits for quick load op code definitions are 0 to #3C.

UNROLLABLE TASK WANTED

Commands Used: PECULIAR

Meaning: This message allows the operator a chance to reconsider.

Recommended Action: Enter '/R' to continue or enter '/A' to abort.

USE THE 'CPD' DIRECTIVE FOR THIS REQUEST

Commands Used: CHANGE

Meaning: TOC/TOC32 no longer supports this directive. It is supported through OC and JM.

Recommended Action: Use the CPD Directive.

WARNING! PGM MAY HAVE WRAPPED AROUND 64K BOUNDARY

Commands Used: CATALOG

Meaning: The program being cataloged exceeds the 64K limit.

Recommended Action: Reduce program size by using overlays or segmentation.

WRONG GEOMETRY OR RECORD SIZE

Commands Used: COMPRESS, LDIRECTORY, LIST, PUT, or RESTORE

Meaning: An End-Of-Record has been encountered in a module while listing. This can mean that the file is not geometry of 1, record size 0.

Recommended Action: Check the geometry and record size parameters of the file.

GLOSSARY

** A **

ABORT
1. To terminate the execution of a task program abnormally. Also called KILL if a task other than the affected task initiates the operation. 2. A REX service or operator directive that causes a task program to be abnormally terminated. As a REX service, it is called by a task that encounters an impossible situation with which it cannot operate.

ACTIVATE
1. To start the execution of a task program. 2. A REX service that a task can call to activate another task. Resident tasks are simply started; non-resident tasks begin memory allocation and loading processes before they are started. 3. An operator directive that causes an inactive task to begin execution.

ACTIVE
A term used to describe the state of a task from the time it has been activated (refer to ACTIVATE) until it terminates execution (exits or aborts). An active task is not necessarily executing; it may have been interrupted by a higher priority task, it can voluntarily relinquish the CPU, suspend, or delay itself.

ASCII
1. American Standard Code for Information Interchange -- also called USASCII. 2. A code used to transmit symbolic characters. 3. A mode of data transmission to and from an I/O device. See Standard ASCII and Nonstandard ASCII.

ASSEMBLER
A low-level programming language processor. The format of Assembler Language statements closely mirrors the hardware structure of the target machine. A typical Assembler Language statement generates a single machine instruction in symbolic form.

ASYNCHRONOUS
1. Not in step with. 2. A term that specifies that an event has no time relationship with another event. 3. Not synchronized in time.

AVFILE
Advance File Directive.

AVRECORD
Advance Record Directive.

** B **

BACKGROUND
1. An obsolete term used to describe only the lowest priority task programs.

BATCH PROCESSING

1. Processing of programs by scheduling them without regard to meeting real-time deadlines. 2. A method of program execution in which control passes from one non-resident chain-overlay program to the next as specified by commands processed by a Job Control program that is loaded as an overlay between each of the other programs. 3. Batch processing is normally performed by one or more tasks designated to execute in one or more fixed partitions. The first such task is usually named B.

BEGINNING OF MEDIUM (BOM)

1. A physical or logical position that defines the start of an I/O device medium. 2. For magnetic tape, the load point immediately ahead of the first (0) record. 3. For disc partitions, a logical position for sequential access that insures that the next forward operation accesses the first relative sector (0) of the partition.

BI

Binary Input File.

BIAS

The amount of memory space skipped by addressing.

BINARY

1. A number system whose radix (base) is two. 2. Data represented by a binary number system. 3. A code used to transmit non-symbolic or transparent data characters, usually eight bits at a time. 4. A mode of data transmission to and from an I/O device. (Refer to Standard Binary.)

BIOS

Basic I/O System.

BKFILE

Back File Directive.

BKRECORD

Back Record Directive.

BO

Binary Output File

BODY

1. The collection of contiguous executable instructions and data areas that constitute a program or task. 2. The core allocated to a task for the loading of continguous executable code and data storage areas. Subsequent overlays cannot exceed this size.

CAN CODE

Stands for Compressed Alphanumeric Code, a 16-bit binary value representing a 3-character symbolic string. It is generated by applying an algorithm to the string that can contain characters only from the following restricted character set, a subset of ASCII:

blank
A - Z
0 - 9
Dollar sign ($), period (.), colon (:)

CATALOGER

A batch processor used to place programs (tasks, overlays, and resource blocks) on a directoried random-access disc partition in load-module format.

CATALOGING

The process of storing object programs or other data in directoried random-access disc partitions. A directory is maintained in a central place (in core or on disc) that allows the cataloged item to be found quickly elsewhere on disc.

CHECKPOINTING

1. The process of writing part or all of a program's body onto disc at various stages of execution. 2. The writing of a fixed partition to disc when needed by a higher priority program. Uncheckpointing is the reading of this area back into memory when the higher priority program no longer needs the area, sometimes called roll-in and roll-out.

CI

Control Input File.

CO

Computer-to-Operator File.

COMMON

A data storage area in memory used by one or more tasks, subroutines, or overlay programs for mutual communications with each other.

COMPLEX DUPLICATES

A duplicate copy of a program that uses different resources than the duplicate master.

CONCURRENT

1. Running together; existing or happening at the same time. 2. A term used to describe the operation of two or more tasks that are active simultaneously even though only one is actually executing.

CONNECT

A REX service or operator directive used to allocate a timer or interrupt for purposes of scheduling the execution of a task and other task control functions.

CONNECTED

A term specifying the scheduled state of a task in the CPU queue awaiting activation (or other control function) by a timer or interrupt.

CONTROLLER

1. A hardware interface operating through a unique I/O channel (group/unit) that accepts control commands and directs them to one or more sub-units (such as a transport, or a pack) of an I/O device. A unique service and data interrupt pair is associated with each controller. 2. A unique data path.

CPU

1. Central Processing Unit. 2. Computer hardware minus memory and peripheral devices. 3. Mainframe.

CSW

Abbreviation for Control Switch. A CPU has 16 such switches that can be used for selecting task and system control functions. These switches are not disabled when the protect system is energized and the control panel is locked.

CYLINDER

An area of disc media that can be addressed without mechanical movement of the disc-head mechanism. If the disc has more than one head, a cylinder represents a number of tracks or revolutions.

** D **

DEVICE

A single-purpose independently-controllable I/O medium. Such a device (sometimes called a logical device) can be a partition of a disc (but not the whole disc) or a magnetic tape transport (but not the entire controller channel).

DIRECTIVE

1. An input record (sometimes typed by an operator) that specifies a mode of operation to be set or a function to be performed by the system or a task. 2. A command used in operator/program communication initiated asynchronously by the operator. Other communication methods, such as a question-and-answer sequence, are initiated by the program rather than by the operator and are not said to involve directives.

DIRECTORY

An ordered data array in memory or on disc that contains pointers and keys to the location of other data items on disc or in memory.

DUPLICATE MASTER

A program that copies are made from.

** E **

EDITOR

A background processor program (overlay) used to link incomplete object programs to external subroutines and data arrays. The result is a complete program called a load module. The Link Editor resolves inter-routine references in one pass over the object code and generates the load module in a second pass. The object code presented to the editor can contain references to routines in one or more subroutine libraries. The editor searches these libraries and incorporates the relevant routines in the generated load module.

EDITING

1. The process of linking an incomplete object program to its called subroutines and data arrays. The process selectively searches subroutine libraries. The result is a complete object program (load module) with all references satisfied. 2. A linking process that produces the linked result as an object module on an external medium--as opposed to link loading, that produces the linked result directly in core.

END-OF-FILE (EOF)

1. A term describing the state entered (a) by an I/O device whenever it encounters a special record while performing input, backspace, or advance operations, or (b) by certain printing devices after printing at bottom of form. 2. (a) A record that if detected causes an I/O device to enter the end-of-file state. Its format is specific to the particular device being referenced. (b) The bottom-of-form position on certain printing devices.

END-OF-MEDIUM (EOM)

1. A physical or logical position that defines the maximum extent of an I/O device medium. 2. For magnetic tape, a marker detectable by hardware, that is located approximately 12 feet from the end of a tape. 3. For disc, a logical status condition that results whenever the last sector of a partition has been accessed.

ESTABLISH

A REX service or operator directive that makes a normally nonresident task resident (its core remains allocated if it exits or aborts) until further notice. If the task is not currently active when it is established, it loads into core and its TCB is entered into the CPU queue.

EVENT

The completion of an operation or the occurrence of a signal that must be communicated to some task. When an event occurs, the Taskmaster is triggered so that it context switches.

EXECUTIVE

1. A collection of programs and data bases with a sole purpose of controlling other programs and to provide services for them. Sometimes called a monitor or operating system. 2. The resident core elements or nucleus of an operating system, sometimes called its core image. 3. The resident nucleus of MAX IV/MAX 32.

EXIT

The REX service a task program can execute at the end of program. For non-resident tasks the memory utilized by the task is freed for other uses and the task becomes inactive.

EXT

Extension linked to a task.

** F **

FAT

File Assign Table.

FCB

File Control Block.

FILE

1. A collection of related records with clearly defined limits, on some I/O medium. The extent of sequential files is usually defined with end-of-file records. The extent of direct access files is defined by a range of valid record numbers. This range information is frequently a part of the file or a related sequential file. Refer to LOGICAL FILE.

FILE MARK

A special record defined in certain data transfer modes that indicates the extent of a file of information. (See END-OF-FILE.) For magnetic tape or disc, a file mark is a special hardware-detected record; for other devices it is a record beginning with two dollar signs ($$) and is detected only in standard ASCII or standard binary formats.

FPI

File Position Index.

FOREGROUND

1. The highest priority group of multiprogram levels in executing in the priority range 0 through 127 and allocating their space from the foreground core pool. 2. A variable pool of core in which foreground tasks and overlay programs execute in areas that are allocated dynamically.

FORTRAN

1. A high-level procedural programming language used to solve engineering and scientific application problems. The name was derived from the initial syllables of FORmula TRANslation. 2. A language processor that translates FORTRAN statements into MODCOMP Macro-Assembler language statements and normally executes as a disc-resident overlay program of a batch-processing task.

** G **

GET

This directive retrieves a task or overlay program already in the current load-module file and puts it in a special object module format.

GLOBAL

1. Being able to be shared by more than one multiprogram level (task). 2. A public memory area, service, or peripheral, that can be concurrently accessed by several tasks. 3. Not local, not dedicated.

GLOBAL FILE

1. A logical file to which I/O operations can be directed from any task; as opposed to a task file, which can only be addressed locally by a particular task. For example, the CO (Computer-to-Operator) file can be addressed by any task waiting to output a message to the system operator. 2. A file belonging to the Operator Communication task and used by it and other tasks. (Refer to GLOBAL.)

** H **

HANDLER

A collection of interrupt driven program elements called by a data or service driver, which controls I/O data transfers (data handler) and the initiation and completion of queued operations to I/O devices (service handler).

HEAD

A transducer in contact with a magnetic (or other) recording medium. A magnetic tape transport has a head in contact with the moving magnetic tape. A moving-head disc has a head for each platter (or surface) of the disc, and the head assembly can move to other discrete media positions on the platter (called cylinders).

HOLD

1. To suspend execution of a task so that only the system operator can resume its execution. 2. An operator directive used to manually suspend task execution. 3. A REX service (option of WAIT and MESSAGE services) that a task can call to put itself into a suspended (hold) state. 4. A state of suspension that only the system operator can release; as distinquished from WAIT, DELAY, or I/O HOLD, that are released by completion of the initiating event or cooperating process.

** I **

ICM

Insert Copiable Module.

IIIALLOCATE

MAX III Allocate Directive.

IMAP

Instruction Map.

INTERACTIVE

A programming method where a computer operator and a computer system closely coordinate a problem solving or control situation, each alternately contributing to the solution. The Operator Communication (OC) task is an interactive program as are many batch processing programs.

INTERRUPT

1. A signal from an external (hardware) or internal (program) source that demands the immediate use of the CPU for another purpose. 2. A program that is executed asynchronously with respect to the program currently executing that interrupts and then returns to the current program.

IOPERATIONS

Input/Output Operations Directive.

ISPACE

Instruction Space Directive.

I/O

Input/Output.

** J **

JOB

1. A unit of program execution for a batch-processing task. 2. A series of one or more program executions scheduled as a unit, sequentially, by the Job Control overlay of the Background Task.

JOB CONTROL
The name of the root overlay program of a batch-processing task. This overlay processes JOB CONTROL statements (or directives) that set up I/O devices and files and then loads and transfers control to other programs on top of itself. It is reloaded automatically when any batch-processing program exits or is aborted.

L

LDIRECTORY
List Directory Directive.

LIBRARY UPDATE
A batch-processing program (overlay) that performs utility copying and editing of lists of sequential object programs and subroutines.

LIFO
Last-In-First-Out

LINK EDITOR
Refer to EDITOR.

LIST
1. A collection of data or programs that are ordered sequentially or linked together with pointers. 2. To print a report on a hard-copy printing device.

LM
The Load Module file is the name of a particular directoried disc file of object modules that is used to hold the highest priority non-resident foreground programs (tasks or overlays). This special file is the only one that can have a core-resident directory as well as a disc-resident directory.

LO
Listing Output File.

LOAD MODULE
A complete object program (task or overlay) in which all subroutines and other references are self-contained or which have been fully linked together by an editing process.

LOCAL
1. Being set apart for use by a single task. 2. Dedicated. (Refer to GLOBAL.)

LOGFILE
Logical File Directive.

LOGICAL DEVICE
1. An independently programmable device or a physical or logical subdivision of a device controller. It can share a queue with other devices; as distinguished from a controller or data path, which can have one or more addressable devices and can have an independent I/O queue.

LOGICAL FILE
An entity with a unique name that a programmer addresses to perform an I/O operation. A file allows a program to be coded in such a way that it is independent of a particular device. A logical device can be assigned (or bound) to the logical file external to the coding of the progam.

LPR
Load program Records.

** M **

MAX
Modular Applications Executive.

MEDIA
1. Plural of medium. 2. The physical space or substance upon which an input/output device stores information: in magnetic tape, the tape itself; in disc, the disc magnetic coating; in cards, the card itself. 3. The extent or amount of physical space an I/O device is capable of utilizing.

MEDIUM
Refer to MEDIA, END OF MEDIUM, BEGINNING OF MEDIUM.

MODULE
1. A segment of object code representing either a complete program or a subroutine, sometimes called an element. 2. A LOAD module is a segment of object code representing a complete program that has no unsatisfied external references and can be loaded and executed by an operating system.

MOR
Module Object Records.

MULTIPROGRAMMING
The concurrent execution of two or more programs on a single computer by means of switching between them. Multiprogramming is used to increase the useful work performed on a computer by taking advantage of the natural delays in programs to allow other programs to operate.

** N **

NESTED
A set or series of things set within the one next larger.

NOVERIFY
Noverify Mode Directive.

** O **

OBJECT
A representation of a program or subroutine, which has already been processed by an assembler or macro assembler, that is in a data format suitable for processing by a loader or editor. In addition to being a specification of machine instructions, addresses, and data arrays it also has information that permits the program to be relocated.

OMAP
Operand Map.

OPERATING SYSTEM
1. A name applied to a collection of hardware or software elements that control, supervise, and provide services for other application-dependent software elements. 2. A monitor or executive.

OSPACE
Operand Space Directive.

OVERLAY
A program or part of a program that can be loaded and executed under the control of an existing task from a disc or other suitable device. An overlay can use resources of the system to the extent permitted by the TCB of the initiating root task.

OVTOC
Overlay TOC.

OVT32
Overlay TOC32.

** P **

PACK
1. A particular physical disc transport mechanism that can be uniquely addressed. Also called a spindle. 2. The physical rotating disc medium, particularly if removable from the transport, also called a volume. 3. To compress several noncontiguous data items into a smaller contiguous space.

PARAMETER
1. A word, number, character string, or other data item that is used as an argument of a macro call, subroutine, or directive. 2. A variable that is given a constant value for a specific purpose.

PARTITION
1. A defined contiguous area of memory for the execution of tasks. 2. Disc partition: a defined contiguous area of disc medium (integral number of tracks) that is given a unique name and made to take on the characteristics of a unique logical device. A partition can be programmed in a manner similar to a magnetic tape transport (sequentially), or it can be randomly accessed by relative sector number (device position index).

PHYSICAL DEVICE
The device transport integrated with its physical recording or transmission medium, as opposed to a logical device to which a programmer assigns the files. A physical device can be divided into many logical devices (such as disc partitions).

PHYSICAL RECORD
A number of words or bytes of data, sometimes called a block, the size is determined by the physical constraints of the device medium; as opposed to a logical record, the size of which is determined by the particular needs of a program.

POOL
1. A collection of available resources. 2. A core pool is a collection of unused core blocks linked together in a queue so that a core allocate can quickly find and remove a particular segment for use by a task. 3. A sequential threaded list of available timers, TCBs or other control blocks.

POPTIONS
Program Options Directive.

PREP
A process of writing hardware control information in each sector of a movable-head disc pack medium. This process writes control bit preambles which identify the track and sector number and is used only by the hardware for synchronization and seek-error checking. Required for new disc packs only.

PRIVILEGED
1. The state of a task that permits it to use any instruction and the ability to write into or branch into areas of memory other than its own body or pool. 2. The state of a area that permits only privileged tasks to modify it.

PROCEDURE
1. A precise step-by-step method for solving a problem or for producing a result. 2. A named symbolic skeleton or prototype that basically defines a step-by-step JOB CONTROL procedure with certain parameters specified as variables. These variables are supplied by parameters of a calling statement ($DO) which specifies the name of the procedure.

PROTECTED
Refer to PRIVILEGED.

** Q **

QUEUE
1. A waiting line of things to do. Queues are used whenever demands are made faster than they can be met. 2. A threaded list of tasks to execute, I/O operations to perform, timers to update, etcetera.

** R **

RECORD
Refer to PHYSICAL RECORD and LOGICAL RECORD.

RECURSIVE
1. The quality of a program element or subroutine (or REX service) that allows it to call itself. 2. The REX services for I/O operations are recursive in that within these services they may have to call themselves to report to an operator the unavailability of a particular I/O device. Recursive qualities are obtained by not modifying any result in memory other than in areas unique to each call (PUSH stack).

RELOCATABLE
The quality of object code which allows a loader to distinguish addresses from instructions and to add relative starting points (bias) to each address when the program is loaded into different regions of absolute core.

RESUME
A REX service or operator directive used to resume the execution of suspended tasks. Same as Release, except that the directive also causes the Operator Communication task to cease accepting further directives.

REX
Request Executive service. A particular unimplemented hardware instruction used as a call mechanism for reentrant system services. The instruction also provides for a controlled transfer from an unprivileged task to privileged state because the execution of the instruction causes the immediate execution of a trap (interrupt).

RMO
Read Module Object.

ROOT TASK
The initial program body executed when a particular multiprogram level is activated. This can be a permanently resident body or an initial overlay program.

** S **

SAC
Loader function code (Set Area Constant).

SC FILE
Scratch File.

SECTOR
1. A physical unit record of a disc. 2. Disc sectors are usually 256 bytes (128 words) in length. 3. The minimum size of a core block (usually 128 words) allocated by the core allocators--sometimes called a page.

SI
Source Input.

SLM
Shared-Load module.

SOURCE
1. A program form as prepared directly by a programmer before it is converted to object format by a compiler or assembler. 2. The symbolic records on which a program is written.

STACK
1. A data table or string of linked data elements whose contents are usually changing on a LAST-IN-FIRST-OUT (LIFO) basis. 2. All available unused resources (for example, I/O nodes, vacant RCBs, etcetera) are linked into stacks or lists. Placing an item in such a stack is called a "push" and removing such an item is called a "pull" operation.

STANDARD ASCII
A mode of transmission to/from an I/O device. In particular, a device-independent transmission mode for symbolic (character) data--as opposed to Nonstandard ASCII, which is a device-dependent mode.

STANDARD BINARY
A mode of transmission to/from an I/O device. In particular, a device-independent transmission mode for binary (non-symbolic) data--as opposed to Non-Standard Binary, which is a device-dependent mode.

SYSPAGES
System Pages Directive.

** T **

TASK
1. A multiprogram level. 2. A program currently operating at a unique multiprogram level including any overlays. 3. A cataloged foreground, middleground or background (fixed-partition) program with its TCB.

TC
Transfer Count.

TMTCB
Two-Map Task Control Block Directive.

TOC
Task Overlay Cataloger.

TOCLOCK
When enabled, this table is used by the module loader and TOC/TOC32 to prevent conflicts between module loads and module file updates.

TRACK
1. A portion of a rotating disc memory that can be addressed during one revolution by one head. 2. A bit channel of a magnetic tape head.

TRANSFER ADDRESS (TA)
1. A dedicated memory cell into which must be placed the starting core address of any I/O operation buffer for a device that is controlled by a Direct Memory Processor (DMP). 2. The address within a program where execution normally begins.

TRANSFER COUNT (TC)
A dedicated memory cell into which must be placed the size of an I/O unit record to be transmitted by a device using a Direct Memory Processor (DMP).

** U **

UNPRIVILEGED
1. The state of a task that does not permit it to use certain machine instructions and confines its influence upon the system to its own body, priority level, or pool. 2. The state of an area that permits any task to modify it (for example, unprivileged global common).

UNPROTECTED
Refer to UNPRIVILEGED.

UPDATE
Refer to SOURCE UPDATE and LIBRARY UPDATE.

** V **

VACANT FILE
A file table entry (six words), either a task file list (in its TWA) or in the system's global file list, which can be assigned for use by a transient or otherwise unknown file name. Such an entry can be freed later and made available for use by another logical file name.

** W **

WEOF
Write End-of-File Directive.

ABORT, 76
access, 57
access right, 51
ACTION Directive, 10, 73
address space, 40
addressing space, 55, 75, 121
Advance File (AVFILE) Directive, 14
Advance Record (AVRECORD) Directive, 15
ALIAS NAME, 34, 81, 84, 100
ALLOCATE Directive, 11
allocated memory, 48
allocations, 45
ARCHIVING, 116
assembler, 113
ASSIGN, 20, 46, 57
ASSIGN Directive, 13
AUTOSTART, 75
AVFILE, 88, 102

Back File (BKFILE) Directive, 18
Back Record (BKRECORD) Directive, 19
batch, 42, 75, 82
batch task, 74
batch-processing, 39
BATCH-PROCESSING TASK, 125
BI, 22, 113
BI file, 27, 86, 102
BIAS, 49, 65, 76, 91
BIAS Directive, 16, 76
BIOS, 76
bit, 68, 78
BKFILE, 88, 102
BLOCK, 40
BLOCK Directive, 20, 54, 91
BO, 14, 42, 113
BO file, 27, 88

CAT, 11, 29
CATALOG, 20, 65, 71, 86, 125
CATALOG Directive, 71
cataloged, 121
cataloging, 57
CATALOGING A TASK, 115
CATALOGING AN OVERLAY, 115
cataloging overlays, 82
catalogs, 3, 22, 119
characteristics, 74
checksums, 102, 144
CI, 63, 113
CLE Directive, 24
Clear Temporary Assignments (CTASSIGNMENTS) Directive, 28

CLMEMORY, 75
CO, 65, 66, 76
CO file, 73
complex duplicates, 48, 144
COMPRESS, 20, 84, 86
COMPRESS Directive, 25, 150
COPY Directive, 27, 42, 88
CPU queue, 100
CTASSIGNMENTS Directive, 63, 82

data chain, 41, 121
data chain length, 20
data chaining, 40, 54
data loss, 40, 46
DEALLOCATE, 48
DEALLOCATE Directive, 29
default, 82
default assignment, 13
DEFAULT Directive, 31
default number, 99
DELAY, 110
DELETE Directive, 33, 84
device, 63
device assignments, 6
device-handler operations, 111
directive mnemonics, 43
directory, 88, 108, 143
DIRECTORY HANDLING, 3
disc space, 86
duplicate, 146
DUPLICATE Directive, 34
duplicate master, 34, 150

End-Of-File, 111
EOF, 27
ERASE, 48
ERASE Directive, 36
execution time, 103
EXIT Directive, 38
EXT, 31, 107
Extended Memory Region, 51

FCB, 99
FILE, 20, 46
FILE Directive, 67, 145
FILE HANDLING, 4
file influence, 97
File Manager, 46
file name, 146
file-management, 122
function code, 149

GAP, 20
GAP Directive, 40, 54

geometry, 39, 46, 155
GET, 113
GET Directive, 4, 42, 80
global, 72
GLOBAL regions, 50
global-shared page regions, 49

HELP Directive, 43
hold, 4
hold state, 10, 66, 73
host task, 103

ID, 31
ID Directive, 44
IMAP, 50, 151
influence, 121, 123
influence limit, 100
influence limits, 69, 78
INFORMATIVE, 5
INITIALIZE, 57, 67
INITIALIZE Directive, 46, 86, 97
initialized, 145
Initializes, 3
INITIALIZING, 114
Input/Output Operations (IOPERATIONS) Directive, 53
INSERT, 22
INSERT Directive, 48, 151
insert types, 43
Instruction Space (ISPACE) Directive, 54
ISPACE, 40, 70

Job Control, 117, 125
Job Control program, 36

KEY Directive, 97

Last-In-First-Out, 7, 95
LDIRECTORY, 61
Link editing, 1
link editor, 113
LIST, 46, 58
List All Entry (LAE) Directive, 58
LIST Directive, 108
List Directory (LDIRECTORY) Directive, 59
List in Order (LIO) Directive, 60
LM file, 52
LO, 113
LO file, 43, 64
load modules, 3, 114
Load Program Records, 121
LOAD TIME, 40, 122
load-module file, 64, 94
load-module files, 121, 122
load-time efficiency, 91

LOGFILE, 6, 28
logical file, 15, 18, 19
Logical File (LOGFILE) Directive, 62
logical file assignments, 82
LOGICAL FILES, 113
lost load time, 40
LPR, 25
LPR space, 41
LPRs, 122, 127

MAP Directive, 64
map output, 127
MASK, 110
MAX III Allocate (IIIALLOCATE) Directive, 45
memory allocation, 45
memory spaces, 54
minimum TC, 40
MODE SELECTION, 4
MODIFY Directive, 65
MODULE HANDLING, 4
Module Object Records, 121
MORs, 122

NAME, 85
new name, 150
noncontiguous, 122
NOTE Directive, 66
NOVERIFY, 46, 97
Noverify Mode, 47
Noverify Mode (NOVERIFY) Directive, 67

OC, 76
OMAP, 50, 151
Operand Space (OSPACE) Directive, 70
operating system, 74
operator directive overlay, 126
operator intervention, 46
OSPACE, 40
overlay, 75, 114
OVERLAY Directive, 71, 81, 147
Overlay Mode, 71
OVT32, 151
OVTOC, 151

page sharing extension, 151
Page-sharing, 31
PAGESHARE Directive, 72, 151
PAUSE Directive, 10, 73
PECULIAR Directive, 74, 86
peculiar types, 43
Peculiarities, 31
PHASE, 110
Pipeline Mode, 75
priorities, 123

priority, 101, 121, 148
Priority and influence limit, 31
private, 72
private pages, 99
PRIVATE regions, 50
privileged, 80, 122, 154
privileged state, 74
Program Options, 31, 43
Program Options (POPTIONS) Directive, 78
push stack, 31
PUT, 117
PUT Directive, 42, 80

quick-load, 39
quick-load format, 121
quick-load op code, 154
quick-load op codes, 147

REASSIGN, 6, 28
REASSIGN Directive, 28, 62, 82
recataloged, 42, 80, 81
RECATALOGING, 118
RECATALOGING A LOAD MODULE, 117
RECATALOGING A TASK WITH NEW RESOURCES, 117
Recatalogs, 4
record size, 39, 46, 155
RELOCATABLE, 76, 151
REMOTE, 76
REMOVE, 75
REMOVE Directive, 33, 84
RENAME Directive, 81, 85
RES, 116
resource, 78
RESOURCE DEFINING, 5
resources, 122
RESTORE, 20
RESTORE Directive, 25, 86
RESTORING, 116
RESUME, 76
Resume I/O Complete Mode, 76
REWIND, 88, 102
REWIND Directive, 87
REX calls, 45
REX service, 78
REX services, 95, 109
RIOCOMPLETE, 76
RMO, 41
root task, 76, 79
root tasks, 68, 78

SAVE, 113
SAVE Directive, 4, 88, 102
scratch file, 26, 94
security key, 123, 148

security lock, 97
separate MORs, 122
shared memory regions, 36, 121
shared pages, 36
shared regions, 48
size limit, 122
SLM, 50
SOURCE, 75
SOURCE Directive, 62, 82, 89
SPACE, 20, 40, 70, 91
SPACE Directive, 11, 54, 91
special symbiont task, 74
SQUEEZE Directive, 94
stack, 99
STACKS Directive, 95
SUBSTITUTE Directive, 57, 97
SYMBIONT, 76
System Operator Communication (OPT) Directive, 68
System Options, 31, 43
System Options (OPTIONS) Directive, 68
System Pages, 31
System Pages (SYSPAGES) Directive, 99

TASK, 81
TASK CONTROL BLOCK, 6
TASK Directive, 100
TC, 54
TCB, 112
TEST Directive, 27, 102
time limit, 103
TIMELIMIT Directive, 103
TMTCB, 81
TMTCB Directive, 80
TOC, 71
Transfer Address (TA), 40
Transfer Count (TC), 40
Two-Map Task Control Block (TMTCB) Directive, 22, 105
two-map TCB, 105
typing error, 82

UNABORTABLE, 76
unallocated space, 48
unprivileged, 25, 123, 154
Unprivileged Mode, 75, 86
UNROLLABLE, 76, 122
USER Directive, 107
user source library, 118

vacant entry, 108
VACATE Directive, 108
VARIABLE Directive, 109
verify, 47, 97
Verify Mode, 67
virtual space, 48

virtual-space regions, 48

WORKBENCH Directive, 112
Write End-Of-File (WEOF) Directive, 111

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310

Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

MODCOMP

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below.

☐

Comments: _____

Manual Title _____

Manual Order Number _____ Issue Date _____

Name _____ Position _____

Company _____

Address _____

_____ Telephone ( ) _____

MCS 148D (1/83)

MODCOMP

Corporate Headquarters
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310. Tel: (305) 974-1380, TWX: 310-372-7637

International Headquarters
MODULAR COMPUTER SERVICES, Inc., The Business Centre, Molly Millars Lane, Wokingham, Berkshire, RG11 2JQ, UK. Tel: 0734-786808, TLX: 851849149

Latin American Sales Headquarters
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310. Tel: (305) 975-6562, TLX: 3727852

Canadian Headquarters
MODCOMP Canada, Ltd., 400 Matheson Blvd. East, Unit 24, Mississauga, Ontario, Canada L4Z 1N8. Tel: (416) 890-0666, TELEX: 06-961279

SALES & SERVICE LOCATIONS THROUGHOUT THE WORLD

The technical contents of this document, while accurate as of the date of publication, are subject to change without notice.

Printed in the U.S.A.